TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Kerman Saapar 184996IADB

Individuaalne ajavõtusüsteem koerte takistusvõistluse treeninguteks

Bakalaureusetöö

Juhendaja: Madis Listak
Doktorikraad
Kaasjuhendaja: Meelis Antoi
Magistrikraad

Tallinn 2021

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: Kerman Saapar

28.04.2021

Annotatsioon

Käesoleva bakalaureusetöö eesmärgiks on luua töötav ajavõtusüsteem koerte takistusvõistluse jaoks, mis mõõdab automaatselt võistleja aega rajal.

Arenduse käigus luuakse nii tarkvaraline kui ka riistvaraline lahendus, mis võimaldab harrastajatel treenida, pidada võistlusi ja analüüsida enda koera sooritusi.

Töö katab arhitektuuri, sõnumiedastusprotokolle, platvormide valikut ja analüüsib olemasolevaid lahendusi turul. Töös räägitakse ka testimisest ja edasiarendustest.

Arenduse tulemuseks on töötav ajavõtusüsteem, mida saab kasutada harrastaja oma igapäevaseks treeninguks.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 46 leheküljel, üheksat peatükki, 17 joonist ja 3 tabelit.

Abstract

Individual timing system for dog agility training

The aim of current thesis is to create a working timing system for dog agility training, which tracks competitor time on the track automatically.

During the development a software and a hardware solution is created, which allows training, create competitions, and analyze results for enthusiasts.

Thesis contains architecture, messaging protocols, choosing the best platforms and it analyzes current products on the market. Testing and further development is also covered.

The result of the development is a working timing system, which an enthusiast can use for everyday training

The thesis is in Estonian language and contains 46 pages of text, 9 chapters, 17 figures and 3 tables.

Lühendite ja mõistete sõnastik

3D	Kolmemõõtmeline
ACK	*Acknowledgement* – signaal andmeedastuses, kui pakett on kohale jõudnud
Agility	Koerte takistusvõistlus
AJAX	*Asynchronous JavaScript And XML* - kogum omavahel seotud veebiarenduse tehnikaid, mida kasutatakse rakenduse kliendi poolel interaktiivsete veebirakenduste loomisel
AMQP	*Advanced Message Queuing Protocol* - progressiivne sõnumijärjekorra protokoll, rakenduskihi protokoll
API	Application Programming Interface - rakendusliides
ARM	*Advanced RISC Machines* – protsessori arhitektuur
Backend	Teenusepoolne keskkond
Bait	Koosneb kaheksast bitist
Bitt	Üht kahendnumbrit mahutav mäluelement
BLL	Business Logic Layer – Äriloogika kiht
Bluetooth	Traadita võrgu protokoll ja tehnoloogia
CSS	*Cascading Style Sheets* - märgistuskeelse dokumendi välisilme kirjeldamiseks
DAL	Data Access Layer – Andmete juurdepääsu kiht
Docker	Tarkvarakomplekt tarkvara konteinerduseks virtualiseerimisega opsüsteemi tasemel
DOM	*Document Object Model* – dokumendi objektimudel
Dongle	Pistikseade
DTLS	*Datagram Transport Layer Security* - protokolli TLS variant, rakendamiseks datagrammiprotokolliga UDP
EMMC	*embedded Multi-Media Controller* – kiip, milles on nii välkmälu kui ka välkmälu kontroller
Frontend	Kasutajaliides, kliendipoolne keskond
Full HD	Kõrglahutusega kuvavorming
GB	Gigabait

Hz	Sagedusühik
HTML	*Hypertext Markup Language* - hüperteksti märgistuskeel
HTTP	*HyperText Transfer Protocol* - hüperteksti edastuse protokoll, rakenduskihi protokoll
IMU	*Inertial Measurement Unit* - mõõtemuundur, mille tajurid mõõdavad joonkiirust ja pöörlemiskiirust
JSON	*Javascript Object Notation* - Javascript'il põhinev andmevahetusvorming
JSX	Eelprotsessor, mis lubab kasutada XML süntaksit Javascript'i koodis
KB	Kilobait
LED	Valgusdiood
M2M	*Machine to Machine* – masinalt masinale
MB	Megabait
MQTT	*Message Queuing Telemetry Transport* - sõnumijärjekorraga telemeetriatransport, lihtsustatud võrguprotokoll
MVC	*Model-View-Controller* – Mudel-Vaade-Kontroller muster
ORM	*Object-Relational Mapping* - objekt-relatsioonvastendus, võimaldab andmebaasis andmeid otsida ja käidelda objektipõhistes keeltes
OSI	*Open Systems Interconnection* - avatud süsteemide sidumise arhitektuur
PCB	*Printed Circuit Board* - trükkplaat
Ping	*packet Internet groper* - interneti pakettkompaja, võrguhaldusinstrument
QoS	*Quality of Service* - teenuse kvaliteet
RAM	*Random-Access Memory* - muutmälu
RFID	*Radio Frequency Identification* - raadiosagedustuvastuse
SSL	*Secure Sockets Layer* - kunagine krüpteerimisprotokoll
TCP	*Transmission Control Protocol* - edastusohje protokoll, transpordikihi (OSI kiht 4) protokoll, protokollil on paketi kohaletoimetuse kõrge usaldustase
TLS	*Transport Layer Security* - protokoll andmete krüpteerimiseks,
UDP	User Datagram Protocol - transpordikihi (OSI kiht 4) protokoll, kasutatakse TCP asemel siis, kui ei nõuta kohaletoimetuse kõrget usaldatavustaset

URI	*Uniform Resource Identifier* - universaalne ressursiidentifikaator
USB	*Universal Serial Bus* – universaalne jadasiin
Wifi	*wireless high – fidelity* – traadita kohtvõrgu tehnoloogia
XML	*Extensible Markup Language* - platvormist sõltumatu märgistuskeel

Sisukord

Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

Agility on maailmas aina kuulsust koguv loomade takistusvõistlus. Vanasti oli kõige levinum hobuste takistusvõistlus, mida korraldati hipodroomidel. Alaga tahtsid tegeleda paljud, aga suurte loomade ülalpidamine on kulukas. Hobuste takistusvõistlusi jälgides tulid koeraomanikud mõttele koertega sedasama teha. Nii sündiski koerte *agility.*

Eestis on *agility* väga menukas ala. Lõputöö autori juurde pöördusid murelikud *agility* harrastajad oma ideega. Nimelt tänapäeval on harrastajad mures, et nende armastatud spordialal ei ole käepärast lahendust ajavõtusüsteemile. Olemasolevad lahendused on kallid, kohmakad ja rasked, nii õppimise kui ka kasutamise seisukohalt.

Käesoleva lõputöö eesmärgiks on luua ajavõtusüsteem treeningute analüüsiks ja abistamiseks harrastajatele. Kasutatakse tavainimesele kättesaadavat elektroonikat ja infotehnoloogilisi vahendeid probleemi lahendamiseks.

Lõputöö koosneb seitsmest peamisest peatükist – probleem, taust, analüüs, süsteemi teostus, testimine, tulemus ja edasiarendused. Esimene peatükk annab ülevaate käesolevast probleemist. Teine peatükk kirjeldab koerte takistusvõistluste ajalugu ja kuidas see Eestisse jõudis.

Lõputöö kolmandas peatükis uuriti turul olevate konkurentide olemasolevaid ajavõtusüsteeme. Selgitati, kuidas koera takistusest läbi hüppamist tuvastatakse ja võrreldi võimalike alternatiividega. Analüüsiti turul olemasolevaid arendusplatvorme ja uuriti milline platvorm sobiks kõige paremini antud ajavõtusüsteemi. Kuna süsteem pidi olema juhtmevaba, siis võrreldi tuntuimaid andmeedastusprotokolle ja reaalajalist sõnumiedastust harrastaja nutitelefonis olevasse klientrakendusse.

Neljas peatükk räägib kuidas kõik komponendid annavad kokku ühe süsteemiterviku, kuidas antud komponente arendati nii riistvaralisest perspektiivist kui ka tarkvaralisest.

Viies peatükk kirjeldab, kuidas viidi füüsiliselt läbi testimist ja kirjeldatakse esinenud probleeme. Kuues peatükk räägib töö tulemustest ja autori saadud kogemustest. Seitsmes peatükk räägib tulevikuplaanidest ja võimalikest suundadest, kuhu edasi areneda.

1.1 Metoodika

Analüüsi osas annab autor ülevaate harrastajate lähtetingimustest. Uurib, millised on turul olevad lahendused ja nende puudused. Võrdleb erinevaid sõnumiedastusprotokolle. Analüüsi tulemusel leiab autor parimad viisid, kuidas antud probleemi lahendada ja millised on parimad töövahendid selle saavutamiseks.

Praktilises osas räägib autor, kuidas probleemi lahendati. Autor annab ülevaate, kuidas süsteem töötab tarkvaraliselt ja räägib kasutatud riistvarast.

Lõpetuseks toob välja edasised arendused.

Kokkuvõttes kirjeldab tehtud tööd, kasutatud lahendusi ja tulemust.

2 Ülevaade probleemist

Treeningutel ja võistlustel kasutatakse suuri tabloosid, mida on päikselise ilmaga halvasti nähtavad. Taolisi tabloosid on tülikas või koguni võimatu transportida treeningutele üksinda. Asjaolu teeb veel raskemaks mitme koeraga treeningule minek. Mitmetel olemasolevatel süsteemidel on toiteallikaks autoaku või tootja poolt tehtud toiteallikas, antud akud ei kesta terve päeva ja vajavad tihti vahetust poole võistluse pealt. Tootja poolt tehtud toiteallikad maksavad palju ja ei ühti teiste süsteemidega.

Turul pakutavad ajavõtusüsteemid ei ole kättesaadavad üksikisikutele. Selle resultaadina peavad mitmed harrastajad ostma ühe süsteemi grupi peale. Gruppideks on tavaliselt kennelliidud. Antud süsteemid on suured ja kohmakad. Juhtmevaba versioon maksab topelt võrreldes juhtmega versiooniga.

Hetkel kasutavad koerte takistusvõistluse harrastajad ajavõtuks telefoni stopperit. See eeldab teise inimese olemasolu ja pole piisavalt täpne, sest iga millisekund loeb raja läbimisel.

Treeningutel soovitakse mõista, kuidas koer saab aru omaniku kehakeelest ja tema käsklustest. Täpse süsteemi olemasolul on võimalik omanikul aru saada, kas suunata koer vasakult või paremalt tõkkest üle ja seeläbi olla kiirem rajal. Harrastajatele annab palju juurde takistuste järjekorra analüüsimine, milliste takistuste vahelisel alal on nende koer kiirem ja millistele aladele tuleks rohkem tähelepanu pöörata, et antud rajalõigu läbimist treenida.

3 Taust

Koerte takistusjooks on lühikese ajalooga spordiala. Antud ala kogub maailmas kiiresti populaarsust. Järgnevas osas tehakse ülevaade *agility* ajaloost.

3.1 Ajalugu

Räägitakse, et *agility* algas vähesest rahast ja tahtest enda koer proovile panna. Paljudel polnud raha osta hobust, et võtta osa hobuste takistusvõistlusest. Seda ei kinnita ametlikud allikad, kuid harrastajatel käib suust suhu selline jutt edasi.

Ametlikult sai koerte *agility* alguse 1970ndatel aastatel, et demonstreerida koerte näituste vaheajal nende kuulekust, osavust ja kiirust [1].
1974 – Inglismaa kuninglike õhujõudude (RAF) koerte üksuse võimekuse ja treeningu demonstratsioon Inglismaa põllumajanduse laadal.
1977 – Peter Meanwellil, koerte treenijal, paluti luua koerte hüppamise üritus pealtvaatajatele Crufts'i koerte näituse vaheajaks. Ta sai palju eeskuju 1974. aastal toimunud põllumajanduse laadal toimunud üritusest.
1978 – Inglismaal Crufts'i koerte näitusel oli üleval esimene takistusrada koertele. Palju inspiratsiooni saadi hobuste takistusvõistlusest. Esimesel rajal olid peamiselt hüpped.
1979 – *Agility* naaseb Crufts'i koerte näitusele. Mitmed koerte klubid Inglismaal hakkasid pakkuma koerte treenimist *agility* eesmärgil.
1980 – *Agility* tunnustati ametlikuks spordiks Inglismaa The Kennel Club poolt koos reeglite ja juhistega. Esimene *agility* võistlus kasutades The Kennel Clubi välja antud reegleid. Samal ajal hakkas *agility* levima ka Ameerikasse.
1984 – Charler „Bud" Kramer avalikustas koerte takistusvõistluse põhitõed ajalehes.
1986 – Loodi Ameerika koerte *agility* ühendus (USDAA) Kenneth Tatschi ja Sandra Davise poolt. Toimus esimene USDAA võistlus Houstonis, Texases 16 koeraga.
1987 - Charles Kramer lõi Rahvusvahelise koerte *agility* komitee (NCDA)

1991 – Koerte *agility* jõudis Eestisse [2]. Toimus „Suomen agilityharrastajien vierailu Tallinnassa“ ehk Soome agilityharrastajate külaskäik Tallinnas. Üritus viidi läbi Tallinna Linnahallis.
2020 – Sügisel pidi toimuma *agility* MM Eestis, kuid COVID-19 tõttu jäeti ära ja lükati järgmise aasta sügisesse.
2021 – Koroonaohu olukord pole muutunud paremuse poole. Eesti Kennelliit otsustas tühistada Eestis sügisel korraldatava MMi. MMi korraldamine antakse edasi järgmisele riigile. [95]

Koerte *agilityst* on veel välja kujunenud näiteks rottide ja jäneste *agility* [22][23].

Joonis 1. Koer agility rada läbimas. Martti Vaidla 2019

4 Analüüs

Analüüs on vajalik osa probleemi lahendamisel. Analüüsi abil leiab parimad lahendusvariandid ja saab luua sügavama arusaamise probleemist. Järgnevas osas analüüsitakse käesolevat probleemi, võimalikke lahendusi ja leitakse parimad tööriistad.

4.1 Lähteülesanne

Harrastaja seab üles raja. Rajal on kuni 22 takistust, olenevalt võistlusklassist. Takistusteks on hüpped, poomid, tunnelid, slaalomid jpm [24]. Vaja on mõõta koera raja läbimise aega stardist finišini kui ka vaheetappe.

Koeral ei tohi reeglite järgi olla ei kaelarihma ega jalutusrihma jooksu ajal. Omanikul ei tohi samuti käes olla mitte midagi jooksu ajal. See elimineerib võimaluse kasutada transpondereid või raadiosagedustuvastuse (RFID) kiipe, et fikseerda väravate läbimist.

Ajavõttu fikseerivaid väravaid (sensorite vaheline ala ehk värav) tohib paigaldada ainult hüpetele. Stardiväravat võib läbida mitu korda. Sensor ei tohi segada, kui koer väravat läbib. Lahendus peab olema juhtmevaba. Lahendus peab olema tehtud laialt levinud elektroonika komponentidest, et edaspidi oleks lihtne remontida ja varuosi leida.

4.2 Olemasolevad lahendused turul

Galican – 3 erinevat ajavõtu süsteemi. [25]

Basic CANometer kaablitega – 30cm*100cm ekraan, 20 meetrit kaablit, 1 laser väraval: 1600€

Advanced CANometer kaablitega – 30cm*100cm ekraan, 20 meetrit kaablit, 2 laserit väraval, rohkem funktsionaalsusi, näiteks 15 sekundi stardikell rajale minekuks: 2600€

Advanced CANometer juhtmevaba – 30cm*100cm ekraan, 2 laserit väraval, 30 tundi aku kestvus: 3600€

Joonis 2. Konkurendi ajavõtusüsteem. Pildilt on näha eraldi toiteplokki ja spetsiaalseid takistuse poste. Martti Vaidla 2020

Tessa Timer

Nõuab eraldi tarkvara paigaldamist arvutisse.

Ühe väravaga süsteem – Suur ekraan, 2 laserit väraval: 444€

Kahe väravaga süsteem – Suur ekraan, 2 laserit väraval: 888€ [26]

Bravedog.lv

Juhtimine puldiga

Kahe väravaga süsteem juhtmevaba – Suur ekraan, 1 laser väraval, 15 tundi aku kestvus: 850€ [27]

Barkr

Ühe väravaga süsteem juhtmevaba – 10 tundi aku kestvus, ekraan värava sensori peal, võimalus aega telefoni näidata. Telefonist süsteemi kasutamine algeline ja kohmakas. 250€ [28]

Joonis 3. Konkurendi toote tabloo. Martti Vaidla 2020

Enamikel turul pakutavate lahenduste abil ei ole võimalik korraldada võistlusi. Nad kõik on ainult eraldiseisvad ajavõtusüsteemid, mis mõõdavad aega punktist A punkti B. Kuid võistluse korraldamine tähendab ka andmetöötlust enne ja pärast koera sooritust. Hetkel toimub see kõik paberil.

Arendatav lahendus hakkab sarnanema Tessa Timer ja Barkri toodetele.

4.3 Värava läbimise tuvastamine

Kuidagi on vaja tuvastada, et koer on läbinud takistuse. Selleks on mitmeid viise.

Kõige levinum on infrapuna laser ja seda registreeriv sensor. Antud sensorid suudavad signaali edastada takistuse tuvastamise hetkest alates 4 millisekundist (250Hz) kuni 0,5 millisekundini (2000Hz). Sellest sõltub suuresti ka sensori hind. [13]

Infrapuna laser ja sensor jaguneb veel kaheks:

- Laser ja sensor ühes korpuses.
- Laser ja sensor eraldi korpustes.

Kui laser ja sensor on ühes korpuses, siis tuleb kasutada reflektoreid.

Kui laser ja sensor on eraldi korpuses, siis tuleb arendada kaks eri seadet väravale, kus mõlemal on eraldi toiteallikas, sest väravate postide vahelisel alal ei tohi juhtmeid olla. Juhtmete olemasolu postide vahel on reeglite kohaselt oht koera tervisele, juhul kui koer jooksu ajal takerdub juhtmetesse.

Ülesande lahendamiseks sobivaid sensoreid kasutatakse tehastes tööstuslike tootmisliinide protsesside sünkroniseerimisel.

Takistuse läbimise tuvastamiseks võib kasutada ka RFID tehnoloogiat [29]. Kui koerale panna kaelarihm RFID kiibiga ja igale postile RFID lugeja, siis saaks igat koera juba jooksu ajal identifitseerida ja operaator ei pea jooksjat käsitsi sisestama. See kõlab mugavamalt kui eelnev lasersensoritega tuvastamine, kuid reeglitest lähtuvalt ei tohi olla koeral mitte ühtegi asja kaelas ega küljes. Enne jooksu peab omanik eemaldama koeralt kaelarihmad, jalutusrihmad ja muu taolise, sest juhul kui koer jääb kinni kaelarihmaga mingi takistuse nurga taha, siis see kujutab ohtu koera tervisele. Seega see tuvastamise viis ei sobi.

Potentsiaali on ka kaameraga tuvastamisel. Kaamerate standard värskendussagedus on 60Hz ja resolutsioon Full HD ehk 1920 korda 1080 pikslit [30]. Antud parameetrite juures toodab kaamera andmeid umbes 160MB minutis ehk 2,6MB sekundis. Mikrokontrolleritel on tüüpiliselt 520KB mälu. Üks pilt ei mahu mällu ja seega ei saa analüüsida, kas koer on takistuse läbinud või mitte. Mikroarvuti lisamine igale väravale tõstab süsteemi hinna väga kalliks. Kui vähendada resolutsiooni, siis värskendussagedus on ikka madalam kui infrapuna laserite võimekus. Odavamapoolsed kaamerad suudavad kuni 240Hz-ni minna, mis on siiski aeglasem kui odav infrapuna sensor. 240Hz-sest värskendussagedusest kiiremini töötavad kaamerad osutuvad juba väga kalliks. Kaameraid saaks kasutada, aga nad toodavad liiga palju müralist infot, mida on vaja töödelda ja seega ei sobi nad väiksesse ja odavasse lahendusse.

Kui vaadata olemasolevaid lahendusi turul, siis kõikidel on kasutusel lasertehnoloogia. Muud tehnoloogiad on oma olemuse poolest keelatud reeglite pärast või muudavad nad kogu süsteemi mõttetult kalliks või suureks. Lasersensorid on ka aastakümneid tehastes testitud ja nad töötavad ekstreemsetes keskkondades. Seega turul olevatel lahendustel on kasutusel hetkel parim tehnoloogia antud probleemile.

4.4 Arendatavad lahendused

Autor planeerib probleemi lahendamisel tükeldada selle mitmeks erinevaks väiksemaks osaks, millest kokku tuleb üks suur süsteem.

Riistvaralahendused:

- Lasersensoritega ajavõtu värav, mis kinnitub takistushüppe postide külge. Värav registreerib koera hüppe kellaaja takistuse läbimisel.
- Keskjaam, kuhu kõik väravad ühenduvad.

Tarkvaralahendused:

- Tarkvara ajavõtu väravale.
- Tarkvara väravate ja keskjaama vaheliseks suhtluseks.
- Teenus, mis käsitleb kõiki päringuid (edaspidi *backend*).
- Klientrakendus kasutades teenusest saadud infot kasutajale kuvamiseks (edaspidi *frontend*).

Ajavõtu värav

Funktsionaalsed nõuded:

- Lasersensor peab omama sisseehitatud wifi ühenduse tuge.
- Lasersensori kiire katkemisel tuleb saata sõnum üle wifi keskjaamale.
- Lasersensoreid peab saama sisse ja välja lülitada üle wifi.

Mittefunktsionaalsed nõuded:

- Peab töötama otsese päikese käes olemist kuni 8 tundi.
- Pritsmekindel.
- Universaalne toiteallikas (akupank).

- Eraldiseisev antenn wifi kaugemaks levikuks.

Keskjaam

Funktsionaalsed nõuded:

- Peab tekitama kohtvõrgu, kuhu väravad ühenduvad.
- Peab tekitama kohtvõrgu, kuhu kliendid ühenduvad.
- Majutama erinevaid konteinereid Dockeris.

Mittefunktsionaalsed nõuded:

- Pritsmekindel.
- Universaalne toiteallikas (akupank).
- Kasutama EMMC mälu mälukaardi asemel.
- Eraldiseisev antenn wifi jaoks.

4.4.1 Võimalikud kasutusvaldkonnad

Antud lahendus on mõeldud koerte takistusvõistlusele, kuid see ei pea piirduma sellega. Ajavõtusüsteemi saab kasutada mitmes valdkonnas. Kõige lähim valdkond on hobuste takistusvõistlus. Hüppe takistused on sarnased ja ajavõttu saaks täpselt sama moodi kasutada ka hobustel. Siinkohal tuleb tähele panna ajavõtusüsteemi sensorite maksimaalset kaugust, sest hobustel on takistused laiemad kui koertel.

Antud lahendust saaks kasutada igal pool, kus võistlus toimub umbes 100x100m alal, näiteks staadionil ja võistleb üksainus võistleja korraga. Staadioni ala limiteerib wifi leviala.

Näiteks kehalise kasvatuse tunnis jooksevad õpilased 60 meetri aega.

Autorallil võib ka antud süsteem kasutust leida. Wifi limiteerib autode ajavõtu alaks umbes 100x100m ala.

4.5 Platvormide valik

Tänapäeval on erinevate platvormide valik suur ja lai. Iga toode pakub erinevaid võimalusi. Järgnevates alampeatükkides on vaadeldud laialt levinud platvorme, mida kasutatakse väikese eelarvega projektides praktikute poolt.

4.5.1 Kontrolleri valik väravatele

Värava kontrolleril peab olema wifi ja eraldiseisev antenn. See peab suutma värava läbimise sensoreid sisse-välja lülitada nendelt andmeid lugeda.

Eraldiseisva antenni põhjuseks on väljaku suurus. Väljaku suurus on vähemalt 20 meetrit korda 40 meetrit. PCB antennid (Printed Circuit Board antenn ehk trükkplaadil olev antenn) ei levi nii kaugele juhul, kui nad on ebasobiva nurga all [31].

Mikroarvuti või mikrokontroller?

Mikroarvuti – näiteks Raspberry PI

Kallis, suur ja vajab palju energiat. Mõeldud rohkem tarkvara haldamiseks, kuid saab ka riistvara edukalt manipuleerida. Tänapäeval suudavad mikroarvutid käitada Dockerit.

Mikrokontroller – näiteks Arduino Uno

Odav, väike ja vajab vähe energiat. Mõeldud riistvara manipuleerimiseks.

Hinnavahe mikroarvutil ja mikrokontrolleril on 20+ kordne.

Mikrokontrolleril on väravate arendamisel mitmed eelised – hind, veakindlus.

Tuntumad mikrokontrollerid (välja on toodud olulisemad parameetrid):

Espressif:

- ESP8266 – Espressifi (ESP mikrokontrollereid tootev firma) poolt loodud 32 bitise protsessoriga kontroller. Toetab wifit. Maksab alla 3 euro. Väga laialt levinud nutikates seadmetes. [5]
- ESP32 – ESP8266 järeltulija. Toetab wifit kui ka BlueToothi. Maksab 5 euro ringis. Turvalisuse peale on väga palju rõhku pandud tootearenduse käigus. [4]

- ESP32-S2 – ESP32 järeltulija. Toetab wifit. Toetab eraldiseisvat antenni. Maksab 7 euro ringis. [32]

Arduino:

- Arduino UNO – 8 bitine protsessor. Ei toeta wifit. Maksab 20 eurot. [6]
- Arduino UNO WIFI REV2 – Suuresti sama nagu Arduino UNO. Toetab wifit. Maksab 38.90 eurot. [7]

STM32:

- STM32WB – 32 bitine ARM protsessor. Toetab wifit ja BlueToothi. Maksab 43 dollarit. [3]

Tabel 1. Mikrokontrollerite võrdlus

Mikrokontroller	WiFi	Eraldiseisev antenn	Järeldus
ESP8266	Jah	Ei	Jäktkuvalt väga populaarne, kuid Espressif ametlikult ei toeta enam antud platvormi.
ESP32	Jah	Ei	Ei toeta eraldiseisvat antenni.
ESP32-S2	Jah	Jah	Katab kõik vajadused.
Arduino UNO	Ei	Ei	Ei toeta wifit.
Arduino UNO WIFI	Jah	Ei	Ei toeta eraldiseisvat antenni ja hind on väga kõrge võrreldes teiste alternatiividega.
STM32WB	Jah	Ei	Hind on väga kõrge võrreldes teiste alternatiividega ja autoril puudub kogemus STM32 perekonnaga ja selle arendusprotsessiga.

Kõik mikrokontrollerid toetavad sensorite sisse-välja lülitamist ja andmete lugemist.

Kõiki funktsionaalseid nõudeid täidab ESP32-S2, samuti on autoril on pikk eelnev kogemus ESP perekonna kasutamisel. Seega osutus antud projektis valituks ESP32-S2 mikrokontroller.

4.5.2 Kontrolleri valik keskjaamale

Eelnevas peatükis on välja toodud mikroarvuti ja mikrokontrolleri erinevused.

Keskjaamal on vaja käitada Dockeri konteinereid. Docker on konteinertehnoloogia Linuxile, millega saab rakenduse pakendada ühte konteinerisse ja see töötab isoleeritult [34][35]. Ehitades mitmeplatvormilise konteineri, on võimalik konteinerit käitada nii keskjaamal kui ka arenduseks kasutataval arvutil. Docker lihtsustab tarkvara tarnet ja vähendab erinevatel seadmetel testimisele kuluvat aega.

Selle nõudega jäävad valikusse ainult mikroarvutid.

Olulisemad parameetrid:

- Dockeri jaoks on vaja 64-bitist protsessorit.
- Vähemalt 2GB RAM'i Dockeriga töötamiseks.
- EMMC mälu. Antud mälu on palju kiirem kui mälukaart ja sellega seoses väheneb võimalus toite kadumisel mälu riknemiseks [36].
- Peab toetama wifit.
- Peab toetama vähemalt kahte USB pesa. Üks teise wifi *dongle*'i jaoks, et tekitada teine kohtvõrk ja teine pesa vajadusel klaviatuuri jaoks.
- Eraldiseisev antenn. Sama põhjus, mis oli eelnevas peatükis välja toodud.

Tuntumad mikroarvutid (välja on toodud ainult olulisemad parameetrid):

Tabel 2. Mikroarvutite võrdlus

Mudel	Protsessor	EMMC mälu	RAM	Wifi	USB pesade arv	Eraldiseisev antenn	Hind
Raspberry Pi 4B [37]	64 bitine	Ei	2, 4 või 8 GB	Jah	4	Ei	66-94 eurot
Orange Pi 4B [8]	64 bitine	Jah, 16 GB	4 GB	Jah	3	Jah	Ladudest otsas
Orange Pi 3 [9]	64 bitine	Jah, 8 GB	2 GB	Jah	4	Jah	55 eurot
Orange Pi PC Plus [18]	32 bitine	Jah, 8 GB	1 GB	Jah	3	Jah	35 eurot
Banana Pi M5 [20]	64 bitine	Jah, 16 GB	4 GB	Ei	4	Ei	60 eurot
Banana Pi M4 [19]	64 bitine	Jah, 8 GB	2 GB	Jah	5	Ei	60 eurot

Nõudmistele vastavad ainult Orange Pi 4B ja Orange Pi 3. Orange Pi 4B on oma võimekuselt suurema EMMC mälu ja RAM'ga. Tegemise hetkel oli Orange Pi 4B ülemaailmselt ladudest otsas. Sellega seoses tuli otsustada Orange Pi 3 kasuks.

4.5.3 Värava programmeerimise tehnoloogia valik

Mikrokontrolleritel valitseb suurelt C keel [56]. Mikrokontrollerite programmeerimise tehnoloogia valik sõltub mikrokontrollerist, mida tahetakse programmeerida. Üldiselt on igal mikrokontrolleril oma tehnoloogia, mida nad toetavad. ESP kasutab enda loodud ESP-IDF'i raamistikku, Arduino kasutab enda loodud raamistikku, STM kasutab enda loodud STM32 raamistikku. Tänapäeval toetavad enamus laialt levinud mikrokontrollerid Arduino arenduskeskkonda.

Arduino – Arduino arenduskeskkonna suureks plussiks on lihtne kasutajaliides. Teeke on kerge juurde integreerida. Kirjutatakse abstrakteeritud C++ keeles. See tähendab, et on tehtud palju mugavusmeetodeid koodi kirjutamiseks ja lõppkokkuvõttes sarnaneb ta teiste kõrgema taseme programmeerimiskeeltega. Isegi kui mikrokontroller toetab Arduino arenduskeskkonda, ei pruugi seal olla kogu tugi kõikidele funktsionaalsustele antud mikrokontrolleril [57].

ESP8266 RTOS SDK – ESP8266 mikrokontrollerite programmeerimise raamistik [58]. Programmeeritakse C keeles. Saab ligi kogu funktsionaalsusele ESP8266 pagasis. Ühel hetkel nähti, et ESP-IDF on mugavam kasutada ja hakati ümber tegema raamistikku ESP-IDF stiili. Nüüdseks on kerge üle minna ESP8266 pealt ESP32 peale.

ESP-IDF – ESP32 programmeerimise raamistik [59]. Hetkel Espressifi poolt toetatud raamistik. Eksisteeris kaua aega koos ESP8266 RTOS SDK raamistikuga.

STM32 – Mõeldud ainult STM32 kontrollerite programmeerimiseks [60]. Väga konfigureeritav. Saab kirjutada nii C kui ka C++ keeles.

Autoril on suur kogemus Arduinoga ja hea kogemus ESP-IDF'i ja ESP8266 RTOS SDK'ga. STM32'ga kogemus puudub. Kuna eelnevalt mikrokontrolleriks oli valitud ESP32-S2 ja antud mikrokontrolleri arendamiseks sobib ESP-IDF. Autor valib mikrokontrolleri arendamiseks tootja poolt soovitatud ESP-IDF raamistiku.

4.5.4 Teenuse programmeerimiskeele valik (*backend*)

Teenuse programmeerimiseks on palju valikuid. Tänapäeval valitseb teenustega liidestamiseks REST API põhimõte.

Tuntuimad raamistikud ja nende programmeerimiskeeled [11]:

Raamistik (Põhiline programmeerimiskeel selle arendamisel)

- Spring (Java) – Avatud lähtekoodiga raamistik. Sobib Javaga kogemust omavatele arendajatele. Alustas enda elu sõltuvuse sisestavuse (Dependency Injection) konteinerist ja seejärel kasvas välja suureks ökosüsteemiks. Pannakse suurt rõhku skaleeritavusele ja on kuulsaim tööriist suurtes ettevõtetes [10].

- Laravel (PHP) – Avatud lähtekoodiga raamistik. Jälgib MVC mustri standardit. Sisaldab paketihaldussüsteemi. Kasutatakse relatsioonilisi andmebaase [39].

- Rails (Ruby) – Tuntud rohkem kui „Ruby on Rails“. Jälgib MVC mustri standardit. Suur skaleeritavus [38].

- Django (Python) – Avatud lähtekoodiga raamistik. Sisaldab enda veebiserverit. Skaleeritav ja kiire [40].

- Express.js (Javascript) – Avatud lähtekoodiga raamistik. Node.js'i kasutab enda kõhus [41].

- ASP.NET Core (C#) – ASP.NETi järeltulija. Toetab mitmeplatvormilisust. Saab käitada nii ARM'il kui ka Intel x86 protsessoritel. Samuti on toetatud enam levinumad erinevad operatsioonisüsteemid - macOS'il, Linuxis kui ka Windowsis. Sisaldab endas veebiserverit nimega Kestrel. Kestrel on üks peamiseid põhjuseid, miks otsustatakse ASP.NET Core kasuks, sest Kestrel on üks kiiremaid veebiservereid hetkel [42].

Kõik välja toodud raamistikud on avatud lähtekoodiga.

Kõik tänapäeval laiemalt kasutust leidvad raamistikud sisaldavad mingil määral ORM'i (Object Relational Mapper) funktsionaalsust, mis automaatselt teisendab andmebaasist tulevad andmed objektideks [61].

Kokkuvõttes üritavad kõik raamistikud samu probleeme lahendada. Andmebaasist infot küsida, seda manipuleerida ja kasutajale anda. Raamistiku valiku otsustus toetub valdavalt arendaja varasematele kogemustele. Autor on kõige tuttavam ASP.NET Core raamistiku ja C# keelega.

4.5.5 Klientrakenduse tehnoloogia (*frontend*)

Klientrakendus ehk *frontend* on koht, kuhu kliendid ühenduvad ja antud lülis kuvatakse kliendi vaated.

Klientrakenduse puhul tuleb arvestada, mis platvormi kliendid valdavalt kasutavad. Näiteks kas neil on mugavam kasutada rakendust platvormipõhiselt telefonis, veebirakendusena või arvutis *desktop* rakendusena.

Telefonile programmeerides tuleb mõelda, kas kirjutada Androidile või iOS'ile [43] [44]. Androidi kodukeel oli vanasti Java, kuid nüüd on selle üle võtnud Kotlin [45]. iOS'ile kirjutatakse Swifti keeles [46]. Otse Androidile või iOS'ile kirjutamise miinus on suur ajakulu. Nad mõlemad on väga komplitseeritud süsteemid ja nendega tuttavaks saamine võtab kaua aega.

Tänapäeval koguvad kuulsust hübriidmobiilirakendused. Hübriidmobiilirakenduste tehnoloogia võimsus seisneb selles, et ühe koodibaasiga saab kompileerida mitmele erinevale platvormile rakenduse. Tuntuimad neist on:

- Flutter – Google'i toode, kirjutatakse Dart keeles [47][48]. Flutter on kui graafikarenderduse mootor, mis joonistab platvormipõhises stiilis nupud, tekstid jms ekraanile. Saab ühe koodibaasiga programmeerida nii Androidile, iOS'ile, Google Fuchsiale, veebi, Linuxile, macOS'ile ja Windowsile. Lühidalt öeldes toetab ta kõiki suurimaid platvorme. Flutter suudab töökindlalt ekraanile joonistada 120 kaadrit sekundis [49].

- Xamarin – Microsofti toode, kirjutatakse C# keeles [50]. Kompileeritakse vastava platvormi baitkoodi, kas Android, iOS või Windows rakenduseks. Toetab NuGeti, mis võimaldab erinevaid valmiskomponente kasutada [51].

- React native – Facebooki loodud, kirjutatakse Javascriptis [52]. Toetab Androidile, Android TVle, iOS'ile, macOS'ile, tvOS'ile, veebi ja Windowsile rakenduste kirjutamist.

Arvutile platvormipõhiselt kirjutades peab platvormi valima sama moodi nagu telefonile platvormipõhiselt kirjutades. Valikus on Linux, Windows ja macOS. Antud

platvormidele tuleb lahendada samad probleemid, mis telefonile platvormipõhiselt kirjutades. Igale platvormile tuleb eraldi tarkvara kirjutada ja see teeb töö väga mahukaks.

Veebi toetavad nii arvutid kui ka telefonid. Veebis kirjutatakse Javascripti, HTML ja CSS segu [53][54][55]. HTML (Hypertext Markup Language) ja CSS (Cascading Style Sheets) pole programmeerimise keeled. HTML on veebilehe selgroo ehk struktuuri loomiseks ja CSS on selle stiliseerimiseks. Javascript on mõeldud veebilehe funktsionaalsuse loomiseks. Kui teenuse programmeerimise platvormi valikul sai valida erinevate keelte vahel, siis klientrakenduste maastikul valitseb enamasti Javascript. Klientrakenduste loomise jaoks on tehtud väga palju raamistikke. Kuulsaimad raamistikud on:

- React – Arendatud Facebooki poolt [62]. Paistab välja oma virtuaalse DOMi manipuleerimise poolest [63]. Komponendipõhine arhitektuur tagab rakenduse skaleeritavuse. Kasutab JSX'i [64]. JSX tundub olevat HTML ja Javascripti segu. JSX on eelprotsessor, milles saab kasutada XML süntaksit [65]. Toetab Typescripti.

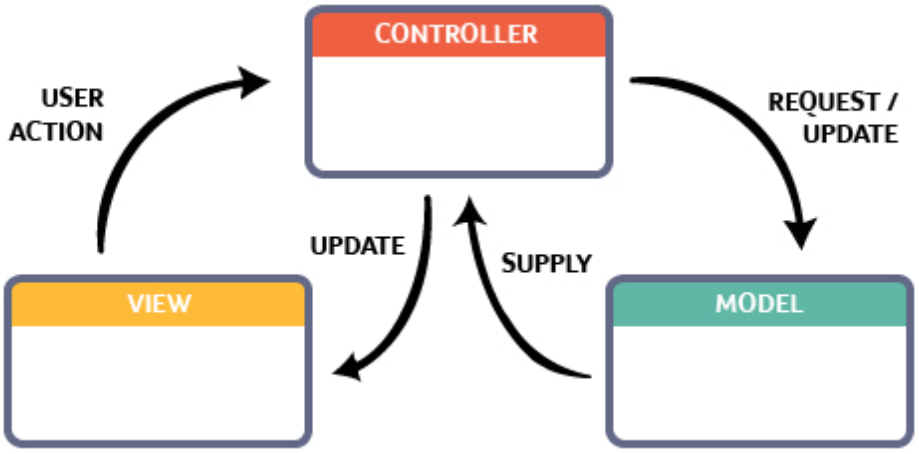
- Angular – Arendatud Google'i poolt [66]. Täissuuruses raamistik. Õppimiskõver on väga järsk. Toetab Typescripti. Kasutab MVC mustrit.

- Vue – Loodud Evan You poolt [67]. Väga väike raamistik mahu poolest. Võeti parimad palad Angularist ja tehti kergekaaluliseks. Mõeldud ühelehe-rakendustele ning ühildub paljude Javascripti pakettidega. Toetab Typescripti. Kasutab MVC mustrit.

- jQuery – Üks esimesi Javascripti raamistikke. Lihtsustab Javascripti kirjutamist [68]. Tänapäeval loetakse seda aeglaseks.

- Aurelia – Täis raamistik. Toetab Typescripti ja sõltuvuste sisestamist (Dependency Injection [70]). Sarnaneb suuresti C# keele kirjutamisele. Väga lihtne kasutada. Kasutab MVC mustrit [68].

Typescript (loodud Microsofti poolt) on edasiarendus Javascriptist [71]. Ta on tugevalt tüübitud programmeerimiskeel, mis transpileerub lõpuks tagasi Javascriptiks.



Joonis 4. MVC muster

MVC ehk Mudel-Vaade-Kontroller (ingl. k. Model-View-Controller) on tarkvaraline arhitektuurimuster, mis jaotab rakenduse kolmeks komponendiks [72][73]. Model ehk mudel on keskne komponent antud mustris. Mudeli tööülesanne on hoida andmeid. View ehk vaade on osa, mida kasutaja näeb ja defineerib ära, kuidas peaks mudelit kuvama. Controller ehk kontroller majandab kõike seda. Kontroller serveerib mudeli vaatesse. Kui kasutaja midagi klõpsab vaates, siis vaade teavitab kontrollerit ja teeb vajalikud toimingud, kas uuendab mudelit või vahetab vaates midagi ümber. Kõik suhtlus käib läbi kontrolleri. Mudel ja vaade omavahel ei suhtle.

Kõik eelnevalt loetletud raamistikud on avatud lähtekoodiga. Kõige populaarsem antud loetelust on React. React on aga pidevas muutumises, mis teeb ta vähem soositavaks antud lõputöös, sest soov on, et süsteem töötaks ka mitme aasta pärast. Angular on suur ja aeglane.

Autor on töötanud 1,5 aastat Xamarini arendajana ja seega on ta näinud hübriidmobiilide rakenduste maailma lähedalt. Hübriidmobiilirakendust pidurdab suuresti arenduse kiirus, sest lahendust tuleb testida mitmel seadmel. Eelnevat kokkupuudet pole autoril olnud Angulariga, seega aja kokkuhoiu mõttes ei ole mõistlik seda nullist õppima hakata. jQuery on väga vana raamistik ja pole tänapäeval mõistlik valik kiire ja skaleeritava rakenduse jaoks. Autoril on aastane kokkupuude Aurelia raamistikuga ja kuna see on täissuuruses raamistik, mis on skaleeritav ja C# keele tundjale lihtne kasutada, siis Aurelia on klientrakenduse kirjutamiseks eelistatud valik.

4.6 Madala latentsusajaga andmeedastusprotokollid

Ajavõtusüsteemis on väga tähtis kiirus. Mida kiirem on andmeedastus, seda täpsemad on tulemused. Andmeedastusprotokoll määrab ära, kuidas väravatega suhelda. Igal protokollil on omad iseärasused ja parimad palad. Tuntumad asjade interneti andmeedastusprotokollid on:

- Message Queuing Telemetry Transport (MQTT) – Üks vanimaid machine to machine (M2M) ehk masinalt masinale protokolle [74]. Loodud aastal 1999. Töötab publish/subscribe tööpõhimõttel [75]. Publish/Subscribe tähendab seda, et seadmed subscribevad ehk kuulavad mingit teemat (topic) ja kui keegi saadab sõnumi sellele teemale, siis näevad ainult need seadmed seda sõnumit, kes kuulavad seda teemat. Teemad on teksti baasil näiteks „temperatuur“ või „valgus“. Vajab keskset serverit kuhu ühenduda, seda nimetatakse MQTT Brokeriks ehk vahendajaks. MQTT on binaarse protokolliga (mitte inimloetav). Sõnum algab alati kindla päisega, et teavitada vastuvõtjat sissetulevast infost. Päiseks on 2 baiti ja sõnumi maksimum suuruseks 256MB. Kasutab TCP transpordi protokolli, et tagada sõnumite kindel kohalejõudmine ja vajadusel TLS/SSL'i turvalisuse tagamiseks [76], [77], [78]. Pakub 3 eri taset Quality of Service (QoS), see tähendab, et saab konfigureerida, kuidas sõnumid kohale tulevad ja kui tähtsad nad on [79]. Näiteks kas on vaja saatjal teada, kas saaja sai sõnumi kätte. QoS 0 ehk tulista ja unusta, saada ja ära jää ootama, kas sõnum saadi kätte või mitte. QoS 1 ehk vähemalt üks kord saadi kätte sõnum. Sellega tekib probleem, et sama sõnumit võidakse saata rohkem kui 1 kord. QoS 2 ehk kindlalt üks kord saadetakse sõnumit. MQTT on mõeldud suurtele võrkudele, mis koosneb väikestest seadetest. MQTT protokoll on väga algeline ja toetab minimaalset funktsionaalsust, see teeb MQTT protokolli väga kergekaaluliseks.

- Constrained Application Protocol (CoAP) – Kergekaaluline M2M protokoll [81]. Mõeldud toimima koos HTTP protokolliga läbi lihtsate käskude [80]. Toetab nii request/response kui ka publish/subscribe tööpõhimõtet. Request/response'i kõige lihtsam näide on HTTP protokollist: näiteks klient pärib info ja saab vastuse tagasi. Teemadeks kasutatakse universaalset ressursiidentifikaatorit (URI) [82]. Tegemist on samuti binaarse protokolliga. Päiseks on 4 baiti. Sõnumid, mida edastatakse on peamiselt väikesemahulised. Maksimum sõnumi suurus sõltub

serverist või kasutatavast programmeerimiskeelest. Kasutab UDP transpordi protokolli ja DTLS'i turvalisuse tagamiseks [83], [84]. Sellega seoses on saadetavad sõnumid kõige väiksemad, aga ei taga sõnumite kohalejõudmist. Toetab kahte eri taset QoS'i. Esimene tase on „kinnitatavad" (CON) sõnumid ja teine „mitte kinnitatavad" (NON). Kinnitavate sõnumite puhul saadab saatja tagasi ACK paketi ja mitte kinnitatavate puhul ei saadeta midagi tagasi. UDP protokolli iseloomust sõltuvalt ei pruugi ACK pakett kohale jõuda saatjale ja sõnum läheb kaotsi. See tähendab, et CoAP'i andmeedastus pole kõige töökindlam.

- Advanced Message Queuing Protocol (AMQP) – Kergekaaluline M2M andmeedastusprotokoll [85]. Toetab mõlemat request/response kui ka publish/subscribe tööpõhimõtet. Vajab keskset serverit, kuhu kliendid saaksid ühenduda. Sõnumeid saadetakse järjekordadesse. Järjekordadel on kindlad nimed. Kui rakendus on sõnumi ära töötlenud, alles siis eemaldatakse sõnum järjekorrast. AMQP on binaarne protokoll. Päiseks on 8 baiti. Sõnumi maksimaalne suurus sõltub serverist või kasutatavast programmeerimiskeelest. Toetab kahte eri QoS taset. „Unsettle Format" ei ole töökindel, see tähendab, et sõnumid ei pruugi kohale jõuda ja „Settle Format" on töökindel tase. „Settle Format"i teeb töökindlaks vastuse saamine sõnumi edukast töötlemisest. Peamiselt on antud protokoll suunatud tööstusele, kus sõnumite kohalejõudmine on oluline ja olemas on lai valik funktsionaalsust.

- Hyper Text Transport Protocol (HTTP) – HTTP protokoll on peamiselt mõeldud veebi päringute jaoks [80]. Toetab ainult request/response tööpõhimõtet. Kasutab universaalset ressursiidentifikaatorit teemade asemel. Tegemist on teksti baasil oleva protokolliga. Ei ütle ette, kui suur peab päis olema. Päise suurus on iga päringuga erinev. HTTP kasutab andmeedastuseks TCP protokolli ja vajadusel TLS/SSL'i andmete turvatud kohalejõudmiseks. HTTP on laialt levinud protokoll veebis.

Kõik välja toodud andmeedastusprotokollid töötavad 7. OSI kihis, sest kõik tuginevad TCP/UDP protokollil [86].

Kõige suurema sõnumi saadab HTTP protokoll, sest antud protokoll pole algselt tehtud asjade internetiks. Kõige väiksema sõnumi saadab CoAP, sest ta töötab UDP protokollil. UDP protokoll on piltlikult öeldes „tulista ja unusta“. Sellepärast ei garanteerita andmete kohalejõudmist. MQTT ja AMQP on UDP'ga selles mõttes sarnased. Mõlemad töötavad TCP protokollil. MQTT sõnum on natuke väiksem, sest päises on 2 baiti võrreldes AMQP 8 baidiga.

CoAP'il on madalaim latentsusaeg tänu UDP protokollile [87], [88]. MQTT on teisel kohal latentsusajaga tänu TCP protokollile. Võrreldes MQTT QoS 1'te ja CoAP kinnitatavaid sõnumeid, siis MQTT saadetud sõnumite latentsusaeg oli natuke suurem kui CoAP'i sõnumil. AMQP sõnumid jõuavad palju hiljem kohale võrreldes MQTT sõnumitega ja HTTP sõnumid võtavad kõige kauem, et kohale jõuda. HTTP sõnumite aeglane kohale jõudmine on tingitud sellest, et sõnumil pole kindlat suurust.

Turvalisus on tänapäeva andmeedastusprotokollidel väga tähtis. Protokollidel puudub turvalisus protokolli tasemel. Turvalisuse tagavad server ja teenus. MQTT broker toetab algelist turvalisuse tuge kasutajanime ja parooli näol [89], [90]. Transpordi kanal toetab TLS/SSL'i kasutamist. CoAP kasutab transpordi kanali krüpteerimiseks DTLS'i. Basic, Digest ja token on laialt levinud HTTP autentimise viisid. AMQP transpordi kanali turvalisuse tagamine toimub TLS/SSL'i abil [89], [90]. AMQP on tööstusele suunatud ja turvalisusele on palju rõhku pandud.

Analüüsides turul kasutavaid protokolle, siis kõige rohkem on kasutusel AMQP teenus/protokoll, sest ta on suunatud tööstusele. HTTP on globaalselt kasutuses veebiteenustes, kuid asjade interneti kasutusalasse ta ei sobi oma paketi suuruse tõttu. CoAP on kõige vähem tuntud, sest pole kasutust leidnud oma UDP transpordi protokolli pärast. MQTT on enim levinud isetegijate hulgas, sest teenus/protokoll on väga kergekaaluline, lihtne aru saada ja kerge üles seada.

Autor valib lõputöösse MQTT protokolli, sest sõnumi kohaletoimetamise tõenäosus ja latentsusaja suhe on parim [87], [88]. TCP protokoll väldib vigu sõnumi kohaletoimetamisel ja analüüsi käigus selgus, et MQTT protokollil on madalaim latentsusaeg TCP kasutavate protokollide hulgast. Lõputöö käigus ei panda rõhku turvalisusele, sest turvalisuse kihid viivad latentsusaja suuremaks.

4.7 Reaalajaline sõnumiedastus klientrakendusse

Eesmärk on kuvada raja läbimise hetke seisu kasutajale veebibrauserisse. Klassikaliselt tuleb on klientrakendused info värskendamiseks sooritavad regulaarselt päringuid serverisse (andmete tõmbamine/pull). Veebibrauserites kasutatakse selleks AJAX päringuid.

AJAX on asünkroonne Javascript ja XML, mis kasutab brauseri sisseehitatud XMLHttpRequest objekti, et serverist andmeid küsida ja Javascripti, et päritud infot kuvada [91].

Arendatavas ajavõtusüsteemis võib igal hetkel tulla uus värava läbimise aeg teenusesse ja kliendrakendus visualiseerib selle alles peale serverist info pärimist. Kui tahta sekundi täpsusega teada uut aega, siis peab 60 korda minutis teenuselt küsima, kas on uus aeg tulnud. Kümne kliendiga tuleks juba 600 päringut minutis. Taoline koormus võib põhjustada teenuse ebastabiilsust.

Tuhandete päringute vältimiseks kasutatakse andmete surumist (push). Ühendust hoitakse kogu aeg lahti kasutades WebSocketit. Tuntuim WebSocketit implementeeriv lahendus on Microsofti toode nimega SignalR.

SignalR teeb kliendi ja serveri vahelise suhtluse kahepoolseks [14]. Server saab kliendile infot saata igal ajahetkel, kui uus info tuleb talle teatavaks. Seda teatakse kui „server push“'i.

SignalR ühendust hoitakse üleval kogu aeg, mitte nagu HTTP päringut, kus iga päringu jaoks ühendutakse uuesti. Ühendus käib üle uue WebSocket transpordi protokolli. Juhul kui WebSocketi protokolli ei toetata või see pole võimalik, siis SignalR klient üritab ühenduda läbi teiste transpordi protokollide [14].

WebSocketit üritatakse kasutada sellepärast esimesena, et tal on [92]:

- Kõige optimaalsem serveri mälu kasutus.
- Madalaim latentsusaeg.
- Kõige rohkem funktsionaalsust k.a täisdupleksühendus serveri ja kliendi vahel.

Kui server või klient ei toeta WebSocketit, siis laskutakse Server-Sent Eventide peale, mis on HTML 5 poolt standardiseeritud. Juhul kui kliendi brauser ei toeta HTML 5 standardit, siis laskutakse Long Pollingu peale.

Long Pollingul hoitakse kliendi päringut senikaua lahti, kuni uus info tuleb. Seejärel saadab server info ära ja klient avab kohe uue ühenduse serveri vastu ja jääb infot ootama [15].

Kõige eelistatuim on siiski WebSockets.

Võimalus on ka kasutada WebSocketit ilma SignalR'ita, kuid SignalR'is on juba paljud murekohad lahendatud. Näiteks vanemate brauserite tugi. Seega on mugavam ja aja kokkuhoiu mõttes mõistlik kasutada SignalR'i.

gRPC on Google'i poolt välja töötatud kaugprotseduuride kõnesüsteem [93]. Ta sarnaneb oma tööpõhimõttel suuresti SignalR'ile. Kasutab sõnumite edastamiseks HTTP/2 protokolli. Sõnumite serialiseerimiseks kasutatakse Protocol Buffereid (Protobuf). gRPC'd kasutatakse suuresti konteinerite vaheliseks suhtluseks. gRPC eeliseks on võimalus kirjutada mitmes eri keeles näiteks Java, C#, C++, Dart, Python, PHP jne.

gRPC-Web toetab brauserisse sõnumite edastamist. Sellepärast on järgnevas tabelis analüüsitud gRPC-Web'i.

Ajavõtusüsteemis on põhilisel kohal kiirus [16]:

Tabel 3. Reaalajaliste sõnumiedastuste toodete andmeedastus kiirus

Tulemused on 10 testi keskmine tulemus.	SignalR	gRPC-Web
Massiiv 100000 elemendiga, 1 päring	00:00.01752	00:00.06181
10 elemendiga massiiv, 1000 päringut	00:00.0079	00:01.0164
Sõne 4194299 tähemärgiga, 1 päring	00:00.004444	00:00.06311
Sõne 10000 tähemärgiga, 1000 päringut	00:00.0085	00:01.05715
Andmete voogedastus	00:08.44667	00:00.07926

Igas testis on SignalR kiirem kui gRPC-Web, välja arvatud andmete voogedastuses. Andmete voogedastust ei rakendata käesolevas lõputöös.

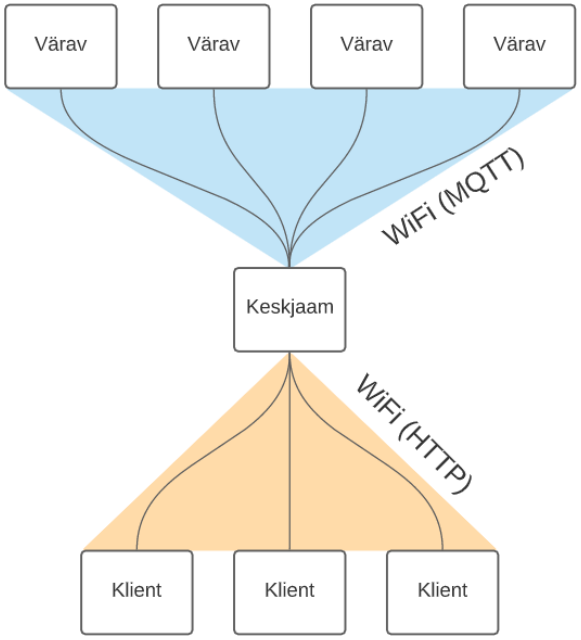
SignalR'i valikut soodustab veel asjaolu, et .NET Core'is saab SignalR'i tööle vähese arendusressursiga. Kogu ülesseadmist ja kasutamist on väga hästi dokumenteeritud.

Autor valib oma lõputöösse reaalajalise sõnumite edastamiseks SignalR tehnoloogia.

5 Süsteem

Süsteem on jaotatud kolme suuremasse peatükki: arhitektuur, riistvara ja tarkvara. Arhitektuuri osas seletatakse lahti, kuidas süsteem kui tervik töötab ja kuidas kõik osised omavahel süsteemi moodustab. Riistvara osas seletatakse lahti, kuidas on värav ja keskjaam tehtud. Tarkvara osas räägitakse, kuidas neli tarkvaralist moodulit on tehtud.

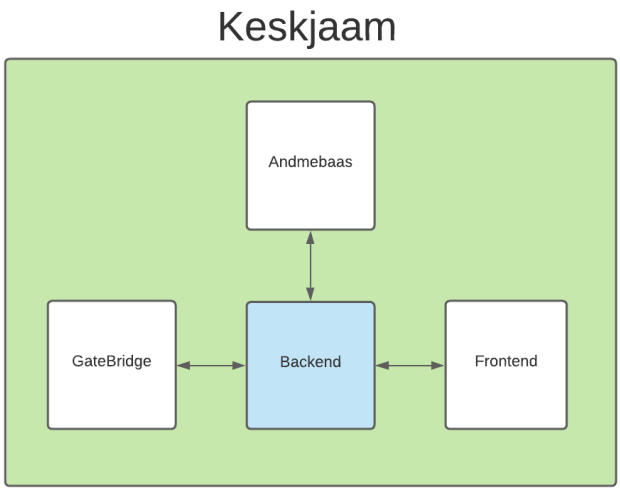
5.1 Arhitektuur



Joonis 5. Topoloogiline ülesehitus

Väravad suhtlevad keskjaamaga üle wifi, MQTT protokolliga. MQTT protokoll sellepärast, et see on kõige levinum madala latentsusajaga protokoll. Väravaid saab ühendada nii palju, kui wifi võrk suudab teenendada, kuid *agility* reeglid sätestavad, et maksimumis on lubatud 14 hüpet rajal. See on suur eelis konkurentide ees, sest konkurentidel on maksimum 2 väravat. Mitu väravat annab võimaluse detailsemalt analüüsida raja läbimist. Ajavõtu väravaid tohib paigaldada ainult hüpetele.

Lõppkasutaja ehk klient suhtleb keskjaamaga üle wifi, HTTP protokolliga.



Joonis 6. Keskjaama komponendid

Keskjaam on kogu süsteemi süda. Arenduse käigus on keskendutud modulaarsusele. Tükke saab vähese vaevaga välja vahetada, kui soov selleks tekib. Kõik tükid keskjaamas töötavad Dockeri konteineris.

Backend vahendab kõiki päringuid ja serveerib vajaminevat infot.

GateBridge on *backendi* ja väravate vaheline suhtlus. See on eraldi tehtud sellepärast, et süsteem oleks modulaarne. Kui tahta kunagi MQTT protokoll millegi muu vastu välja vahetada, siis on see selle tüki välja vahetamisega väga lihtne.

Frontend on kasutajaliides, mida lõppkasutaja näeb.

Andmebaas talletab kogu infot ja serveerib seda *backendile*. Andmebaasiks on valitud MariaDB, sest see on tasuta ja ei sea suuri piiranguid võrreldes MS SQL'i või MySQL'ga.

5.2 Riistvara

Riistvara on loodud kasutades laialt levinud elektroonikat ja arendusplaate. Järgnevates alampeatükkides on kirjeldatud, kuidas antud süsteemile riistvara loodi.

5.2.1 Värav

Värava ajuks on ESP32 arendusplaat, millele on liidetud juurde eraldiseisev antenn.

Sensoriteks on kolm silindrikujulist E3F-R2N2 fotoelektrilist sensorit. Tegemist on infrapunalaser sensoritega. Infrapuna on antud kontekstis vajalik sellepärast, et ta ei sega koera nägemist [12]. Antud sensorid vajavad valguskiire tagasi põrkumiseks reflektorit. Autor kasutab reflektoriks tavalist jalgratta helkurit. Ühe sensori hinnaks on umbes 10 eurot.

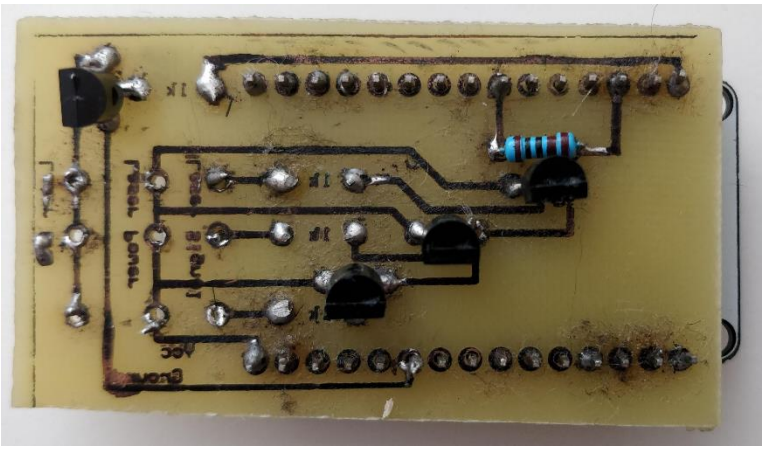
Joonis 7. E3F-R2N2 sensor.

Tihti on harrastajatel kaebused, et ajavõtusüsteem ei registreerinud koera hüpet. Mõni koer võib kõrgemale hüpata, mõni madalamale. Turul olevatel lahendustel on tihti üks või kaks sensorit. Autor paigutas kokku kolm sensorit 15cm vahega, et katta suurem osa koera hüppest. Kolm sensorit peaks selle probleemi lahendama.

Sensorid tekitavad pinge muutuse kaudu signaali kontrollerile, kui sensorisse tagasi peegelduv laserkiir katkeb. Kõikide sensorite signaalid on liidetud üheks signaaliks, sest ei ole vaja teada, millisest sensorist on signaal tulnud. Signaal on liidetud üheks OR loogikaga. OR loogika on tehtud BC337 transistoridega.

Trükiplaate tehti kahes iteratsioonis. Algselt tehti kolm trükiplaati kodustes tingimustes.

Esimesel iteratsioonil kavandati trükkplaadi disain programmis nimega KiCad. KiCad on tasuta saadaolev programm, mis lihtsustab elektriskeemide disainimist ja skeemidest trükkplaadi koostamist. Esimene iteratsioon tehti kodustes tingimustes [17]. Trükkplaadi disain prinditi fotopaberile. Seejärel osteti ühepoolne trükkplaat. Suurest trükkplaadist lõigati välja sama suur tükk nagu välja prinditud trükkplaadi disain. Fotopaber pandi pilt allapoole välja lõigatud trükkplaadi tükile ja seejärel kuumutati triikrauaga. Fotopaber jättis oma kujutise trükkplaadile. Viimane samm ei tulnud esimese korraga välja ja seda tuli korrata mitu korda, sest fotopaber ei jätnud tervet joonist trükkplaadile. Mõned rajad sai parandada veekindla markeriga. Järgnevalt pandi trükkplaat raudkloriidi ($FeCl_3$) sisse, et söövitada üleliigne vaskkate ära. Fotopaberilt tulnud jooned jäid alles, sest raudkloriid ei pääsenud joonise alla ja joonis jäi ilusti nähtavaks koos vasega. Nüüdseks jäid ainult veel augud puurida. Augud puuriti Dremeli-nimelise väikese puuriga. Pärast aukude puurimist oli trükkplaat valmis ja seejärel joodeti kõik komponendid trükkplaadile.



Joonis 8. Esimese iteratsiooni trükkplaat joodetud komponentidega (pildilt on puudu sensorid).

Pärast esimest iteratsiooni sai autor aru, et antud lahendus pole piisavalt modulaarne ja seda masstoota kodustes tingimustes on väga keeruline ja aeganõudev. Lahenduseks oli tellida valmis trükkplaadid.

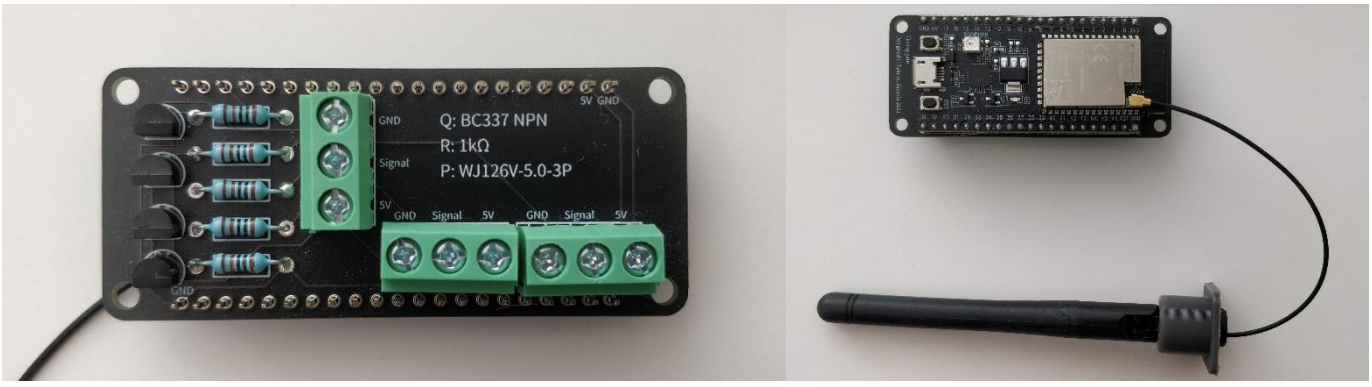
Teine iteratsioon trükkplaate tehti programmiga, mille nimi on EasyEDA. Programm vahetati välja sellepärast, et KiCadi oli väga tülikas uusi komponente sisse laadida.

Teisele iteratsioonile tehti mitu parandust:

- Pandi sarnased komponendid üksteise lähedale, et järjest oleks lihtsam neid joota.
- Eelmisel iteratsioonil olid sensorid trükkplaadile joodetud. Nüüd pandi sensorite ühendamiseks klemmiplokid, et sensoreid oleks lihtsam külge panna ja vajadusel välja vahetada.
- Trükkplaadile pandi legend, mille järgi on lihtsam aru saada jootmisel, milline komponent kuhu käib.

Trükkplaadid telliti Hiinast firmast nimega JLCPCB. Tegemist on ühe suurima Hiina trükkplaadi ettevõttega. Hiinat eelistati sellepärast, et trükkplaadi tükihind oli väga odav. Ühe trükkplaadi hinnaks kujunes 1 euro ja 7 senti. Kokku telliti 20 trükkplaati. Tellitud kogus oli suurem sellepärast, et väiksema kogusega küündis ühe trükkplaadi hind 3 euroni. Mida suurem kogus, seda odavam oli trükkplaadi hind.

Peale trükkplaatide kätte saamist joodeti komponendid trükkplaatidele ja esimese iteratsiooni plaadid asendati teise iteratsiooni plaatidega.



Joonis 9. Teise iteratsiooni trükkplaat joodetud komponentidega koos antenniga (sensorid ühendatakse roheliste klemmiplokkide külge)

Algselt olid väravad tehtud kasutades statiivi ja puulatti, millesse oli puuritud kolm auku sensorite jaoks. Puulatt oli kruvitud statiivi külge. Peale esimest testimist väljakul selgus, et see on liiga kohmakas ja ei sobi mitte kuidagi. Selle parandamiseks hakati mõtlema, kuidas teha väiksemat ja lihtsamini paigaldatavat väravat.

Joonis 10. Statiivil olev prototüüp

Hiinast telliti otse tehasest anodeeritud alumiiniumist kestad. Hiinat eelistati sellepärast, et otse tehasest tellides on sadu erinevaid disaine, mille seast valida. Kestal on mõlemas otsas avad. Kesta külgedele puuriti kolm avaust sensorite jaoks. Ühe kesta hinnaks osutus 30 eurot koos transpordi tasu ja tollimaksuga. Kogu riistvara mahutati kesta sisse. Tulemuseks on vastupidav ja kerge kest, mis kannatab kukkumisi ja näeb esteetiliselt korrektne välja.

Väravatele kinnitamine lahendati tavalise takjapaelaga. Takjapaela eelis seisneb selles, et seda on lihtne paigaldada ja ei vaja spetsiaalset kinnituskohta takistuse postil.

Joonis 11. Osaliselt komplekteeritud värav

5.2.2 Keskjaam

Keskjaamaks on Orange Pi 3. Antud arvutil on piisavalt jõudlust, et infosüsteemi kõiki komponende käidelda.

Keskjaamale on paigaldatud sama antenn, mis väravale.

Orange Pi 3 on pandud 3D prinditud kesta sisse, et vältida selle märjaks saamist halva ilmaga.

5.3 Tarkvara

Tarkvara arendades on suurt rõhku pandud modulaarsusele. Modulaarsuse eeliseks on võimalus arendada ja testida ning hiljem ka muuta ning hooldada tarkvara väiksema ajakuluga. Projekti saab samaaegselt arendada mitu arendajat tulevikus ja ei takistata üksteist. Järgnevates alampeatükkides on kirjeldatud tarkvara arendusprotsessi ja arhitektuuri.

5.3.1 Värav

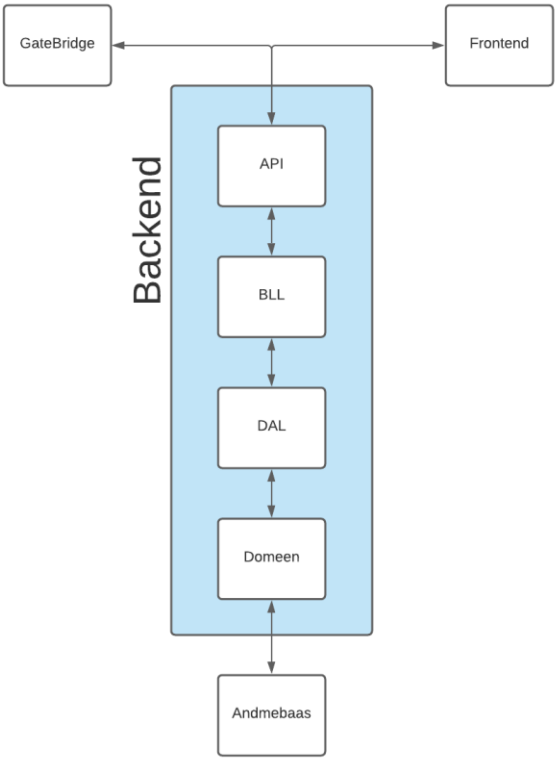
Värava tarkvara on kirjutatud C keeles ESP-IDF raamistikku kasutades.

Kui värav sisse lülitatakse, üritab ta ühenduda ette antud wifisse. Kui see õnnestub, siis üritab ta ühenduda MQTT vahendaja külge. Juhul kui kumbki neist ebaõnnestub, annab värav teada sellest punase LED tulega.

Õnnelikul ühendumisel pingib keskjaam väravat, et teada saada värava keskmist vastamise viidet. See arvutatakse igal väraval võistluse ajal sissetulnud ajast maha. Juhul kui vastamise aeg on ebanormaalselt suur, kästakse värav uuesti seadistada. Põhiline probleem selle ilmnemisel on kehv antenni asetus. Pingimist korratakse iga jooksu alguses, sest jooksude vahel võib-olla juhtus midagi väravaga.

5.3.2 Teenus (*backend*)

Teenus on jaotatud kihtideks. Kihtideks jaotamine tagab rakenduse modulaarsuse.



Joonis 12. Teenuse arhitektuur

Kõige madalamas kihis on domeeni kiht. Domeenis on domeeniobjekte väljendavad klassid.

Andmete juurdepääsu kiht ehk Data Access Layer (edaspidi DAL) on repositooriume haldav kiht. Repositooriumid võimaldavad taaskasutada meetodeid kõrgemates kihtides andmete pärimiseks.

Äriloogika kiht ehk Business Logic Layer (edaspidi BLL) on äriloogikat haldav kiht. Andmetega manipuleerimine ja nende töötlemine toimub BLL kihis. Äriloogika eraldihoidmine annab võimaluse kirjutada mitu erinevat API versiooni taaskasutades äriloogika koodi.

Rakenduse liides ehk Application Programming Interface (edaspidi API) on liidestuse punkt teiste rakendustega. See võimaldab suhelda erinevatel programmidel. API on *backendi* sisendpunkt. Selle vastu tehtavad päringud liiguvad mööda kihte alla kuni andmebaasini või olenevalt selle kihini, mida teha tahetakse.

Frontend ja GateBridge on liidestatud API külge.

Rakenduse API kihis on kasutatud lisaks veel SignalR toodet.

5.3.3 Väravate ja teenuse vaheline suhtlus (GateBridge)

GateBridge on väravate ja teenuse vaheline lüli. Ta on eraldi sellepärast, et kui kunagi on soov mingil teisel protokollil suhelda väravatega, on selle lüli välja vahetamine lihtne.

GateBridge loob MQTT Brokeri serveri ehk tema külge ühenduvad kõik väravad. Broker ehk vahendaja saab monitoorida kõike, mis läbi tema käib. Kõik sõnumid, kõik teemad on hallatavad. Võimalus on ka ühendusi või saadetud sõnumeid blokeerida.

Värava läbimisel saadab värav sõnumi keskjaamale. Keskjaamas loeb väravate sõnumeid GateBridge. GateBridge saadab päringu edasi teenuse liidestuspunkti. Päringu andmeteks on värava identifikaator, kes sõnumi saatis ja hetke aeg läbimise aja registreerimiseks. Sellest sammust edasi haldab kõike teenus.

GateBridge on piltlikult öeldes tõlkija, mis tõlgib vajaminevad sõnumid MQTT protokollist HTTP protokolli ja vastupidi.

5.3.4 Kasutajaliides (*frontend*)

Kasutajaliides tehti veebibrauseritele. Veebibrauserit saab igas seadmes kasutada ja selle testimine on lihtsam. Javascripti raamistikuks osutus analüüsi tagajärjel Aurelia.

Aurelias järgitakse MVC (Model-View-Controller) mustrit.

Mudeliks on teenuse liidestuspunktile vastav objekt. Kontrolleriks on Typescripti fail, mis haldab kogu elutsüklit. Vaateks on HTML fail.

Kogu klientrakendus on kirjutatud Typescriptis.

Klientrakenduses on üritatud hoida samasugust joont nagu teenuses. Domeenis on teenuste liidestuspunktidele vastavad objektid, mida teenus välja annab. Teenusest tulev JSON tõlgendatakse ümber domeeniobjektiks. Repositooriumitesse on kogutud kokku ühe domeeniobjekti kohta erinevad toimingud, näiteks lisamine või uuendamine jms. Kontroller kasutab antud repositooriume.

SignalR kasutamine

SignalR'i kasutamiseks on vaja nii serveris kui ka klientrakenduses arendada sellele vastav tugi. Klientrakenduses on selle rakendamine lihtne.

Peab alla laadima @microsoft/signalr ja @types/node Javascripti teegid. Teenusega ühendumiseks on vaja luua HubConnection.

```
this.connection = new HubConnectionBuilder()
    .withUrl("https://" + window.location.hostname + ":" + environment.port + "/timehub")
    .withAutomaticReconnect()
    .configureLogging(LogLevel.Information)
    .build();
```

Joonis 13. SignalR ühenduse loomine.

Joonisel olev this.connection on HubConnection tüüpi. HubConnectionBuilder tagastab HubConnectioni. URL ütleb, kuhu peab ühenduma. Antud näites on ühendatud localhosti külge. Järgnevalt kasutatakse this.connectioni toimingute tegemiseks.

```
this.connection.on("TimeMessage", (serverTime, obstaclePassTime) => this.
newTimeMessage(servertime, obstaclePassTime));
```

Joonis 14. Sõnumi saabumisel käivitatav meetod.

Koodinäites 2 on välja toodud, mida tehakse, kui saadetakse „TimeMessage“ teemale sõnum. Antud näites antakse 2 muutujat „serverTime“ ja „obstaclePassTime“ edasi meetodile nimega „newTimeMessage“.

```
public newTimeMessage(servertime: string, obstaclePassTime: string) {
    var newTime: INewTime = {
        serverTime: servertime,
        obstaclePassTime: obstaclePassTime
    }
    this.ea.publish("NewTime", newTime);
}
```

Joonis 15. Sõnumi käsitlemine.

„newTimeMessage“ meetodis luuakse uus objekt vastavalt sisse tulnud andmetele. Edasi saadetakse see üle Aurelia projekti laiali, kasutades EventAggregatorit. EventAggregatorit kasutatakse siis, kui on vaja teavet saata rohkem kui ühele saajale. Sõnumi saatmine on sarnane MQTT protokollile. Tal on topic ehk teema ja payload ehk teave, mida tuleb saata. Saajad kuulavad ainult vajalikke teemasid.

```
this.timerSignalRService.start().then(_ => {
    this.ea.subscribe("NewTime", (data: INewTime) => this.newTimeMessage(data));
}).catch(_ => {
    this.failedToStart();
});
```

Joonis 16. Kontrolleris EventAggregatori sõnumitele registreerumine.

Kontrolleris tuleb registreerida EventAggregatori sõnumeid kuulama. Sarnaselt koodinäide 2. Kontrolleris oleva „newTimeMessage“ meetodis saab juba vaadet muuta olenevalt andmetele.

Seadmete kellade erinevus

Testimise käigus selgus, et erinevate seadmete kellad ei ole kunagi ühesuguse täpsusega kuna nad kõik on veidi erinevalt valmistatud.

Antud ajavõtusüsteemis pole vaja kuvada värava läbimise täpset aega, vaid mõõdetakse ainult aegade erinevusi. See nõue lihtsustab suuresti arendatavat süsteemi, sest keskjaama kell ei pea olema täpne.

Et kõik seadmed näitaks ühist kella, on neil vaja referentsaega. Referentsajaks on võetud keskjaama kell.

Iga uue takistuse läbimise vastusele saadetakse kaasa keskjaama referentsaeg, et klient teaks keskjaama aega. Tänu sellele saab klient välja arvutada kellade erinevuse ja kompenseerida vea.

```
public newTimeMessage(newTime: INewTime) {
    clearInterval(this.timerInterval);

    let startTime = new Date(Date.parse(newTime.serverTime));
    // Compensate device clock error
    let error = Date.now() - startTime.getTime();

    // If there is active run time, replace startTime with active run time
    if (newTime.obstaclePassTime !== undefined) {
        startTime = new Date(Date.parse(newTime.obstaclePassTime));
    }
    this.timerInterval = setInterval(() => this.timerIntervalLoop(this.timeText, startTime, error), 10);
}
```

Joonis 17. Kella erinevuse vahe välja arvutamine ja intervalli seadmine.

6 Testimine

Testimas käidi arendusprotsessi käigus, mitte ainult lõpplahendusega. Seega kajastub testimise protsessis ka eelnevaid lahendusi ja kuidas neid paremaks tehti.

Süsteemi testiti füüsiliselt *agility* treeningväljakul. Treeningväljakud on klubidel kas renditud või ostetud. Suvel asuvad treeningväljakud õues, kehvemate ilmadega minnakse sisehalli. Lähim väline treeningväljak asub Tallinna Teletorni taga Kiviaia teel. Lähim *agility* sisehall asub Loos, Lepa tee 2a.

Esimene testimine ei lähe tavaliselt nii nagu peab. Esimesel korral selgus, et klientrakenduses olid sisse jäänud välised veebilehed. Arenduse käigus oli keskjaam kogu aeg internetiga ühendatud, sellepärast ei satutud sellise probleemi ei tuvastatud eelnevalt. Väravate ülespanek oli natuke kohmakas. Reflektorite paigaldamine tekitas probleeme, sest ei saadud täpselt aru, kas sensor tuvastab laserkiirt või mitte. Esimese testimise järeldused olid:

- Kõik peab lokaalselt töötama, mitte midagi ei tohi internetist alla laadida, sest keskjaamal ei pruugi internetti olla.
- Reflektorite ja laserkiire seadistamisel on vaja teada, kas joondus on korrektne. Seda tuleb teada anda paigaldajale visuaalse tulukesega, sest rakendusest joondust kontrollida on väga tülikas.
- Statiivil olev esimene prototüüp ei sobi välikasutusse, sest Eesti ilmastik on ettearvamatu ja igal hetkel võib vihma sadama hakata. Väike vihm võib lahtise elektroonika hävitada.

Peale esimest testimist hakati välja töötama lahendust, kuidas panna kogu elektroonika kinnise kesta sisse, et loodusjõud ei saaks ligi haprale elektroonikale. Sellest on eelnevalt kirjutatud värava riistvara peatükis.

Joonduse kontrollimiseks hakati kasutama arendusplaadil olevat LED tuld, et anda paigaldajale teada joonduse staatusest.

Arenduse käigus tuleb edaspidi rangelt jälgida, et alati kasutatakse lokaalseid faile ja teeke. Vastasel juhul juhtub sama, mis eelneva testimise käigus ja süsteem ei käivitu.

Teisel testimisel saadi teada, et Dockeri töös võib esineda mitmesuguseid häireid. Testimine jäi lühikeseks, sest sellel hetkel Dockeri viimasel versioonil ei suutnud Orange Pi konteinerid käitada. OrangePi ei suuda kompileerida Dockeri konteinereid iseseisvalt ja protsess jääb seisma, jõudes teatud sammuni. Lahenduseks sai arendusarvutis mitmeplatvormilise konteineri ehitamine. Konteiner laeti Docker hub'i ja sealt laeti konteiner alla OrangePi'le. Nii õnnestus OrangePi'l konteinerid edukalt tööle saada.

Kolmandal testimisel leiti mitme klientseadmega testimisel, et iga seade näitab erinevat aega. Kõik kellad pole tehtud sama moodi ja iga seadme kell näitab kas paar sekundit ette või taha. Harv juhus, kui kell on täpne. Selle probleemi lahendust on lahti seletatud peatükis nr 5.3.4 „Kasutajaliides (*frontend*)" lõigus „Seadmete kellade erinevus".

Aja mõõtmise veamäär on hetkel teadmata. Testide käigus on selgunud, et süsteem on palju täpsem kui käsitsi stopperiga aja võtt. Peale esimese versiooni valmis saamist on plaan pöörduda sertifitseerimise asutuse poole ja saada mõõtevea hinnang, mis annaks konkurentide ees suure eelise, kuna mõõtetäpsust pole konkurentide seadmetel välja toodud. Sertifikaat annaks kindluse, et süsteem on täpne ja annaks konkurentide ees suure eelise. Eestis tegeleb sertifitseerimisega Metrosert AS.

Testimine on väga tähtis osa süsteemi arendamisel ja aitas leida väga palju puudujääke, mida arendamise ajal ei nähtud või oli isegi võimatu ette näha.

7 Tulemus

Lõputöö käigus arendati prototüüp ajavõtusüsteem, mis rahuldab suurema osa püstitatud eesmärkidest. Põhilised eesmärgid on täidetud, arendamist vajavad veel näiteks klientrakenduses võistluste korraldussüsteem ja üldine süsteemi haldusliides. Arendatud süsteem kahe väravaga maksab umbes 250 eurot. Hinnale pole juurde arvestatud arenduskulusid. Võrreldes konkurentide ajavõtusüsteemidega on arendatud lahendus kordades odavam. Barkr süsteem on samas hinnaklassis, aga ei sisalda mitmeid vajalikke funktsionaalsusi. Arendatud süsteem on kokkuvõttes funktsionaalsem kui konkurentidel. Lõputöö esitamise ajal ei ole veel jõutud müügikõlblikku staadiumisse.

Arendusprotsessi käigus puutus autor kokku elektroonikaga, sensoorikaga, programmeerimisega, 3D printimisega kui ka disainimisega. Kõikides valdkondades on autor suuresti arenenud ja annab enesekindlust, et tulevikus nendes valdkondades veel rohkem süvitsi minna.

8 Edasiarendused

Edasiarenduse võimalusi on mitmeid.

Ametlikel võistlustel on koertel aega 15 sekundit, et startida. Võimalus oleks juurde arendada kas Bluetoothiga töötav eraldi kõlar või laiendus olemasolevale väravale, mis annab helisignaali 15 sekundit enne starti.

Probleemiks on maha aetud väravate avastamine. Seda praeguse lahendusega tuvastada ei saa. Võimalus oleks laiendada väravat IMU (Inertial Measurement Unit) kiibiga [94]. Sellega saaks tuvastada, kas värav on liikumises ja anda keskjaamale teada kohe, et värav on maha kukkunud ja läbimist lugeda veaks.

Praeguses süsteemis pole lahendatud olukorda, kui keskjaam ei vasta päringutele. Juhul kui väravate sõnumid ei jõua kohale, siis ei saa ka aega kinni panna. Selle mure lahendaks eraldiseisev kell väravatel. Jooksu alguses sünkroonitaks kõikide väravate kellad keskjaama kellaga. Juhul kui keskjaam ei vasta väravatele, siis väravad saavad sõnumid ära salvestada ja uuesti proovida keskjaamale saata. See suurendaks süsteemi töökindlust.

Klientidele, kes soovivad siiski suurt ekraani jooksu aja nägemiseks, võib arendada ekraani, mis liidestub ajavõtusüsteemiga. Antud ekraan peaks järgima samu põhimõtteid, millega on arendatud värav ja keskjaam, näiteks veekindlus ja akupangaga ühtivus. Ekraani saaks moodulitest ehitada täpselt sama moodi nagu on suured reklaamtahvlid ehitatud.

Tulevikus on suur soov liidestada loodud ajavõtusüsteemi genereeritavad raportid mingite teiste olemasolevate süsteemidega: näiteks andmebaasid, kus on koera kõik võistlused kirjas, ilma et kasutaja peaks neid ise käsitsi sisestama.

Kuna suur rõhk on antud töös pandud modulaarsusele, siis edasiarenduse mõttes oleks jätkusuutlik teha ka sensorid modulaarseks. Võimalus sensor välja vahetada annaks võimaluse kasutajal valida, kui laiale takistusele värav pannakse, või kui vähendada kolmelt sensorilt ühele, siis aku kestaks kauem. Antud lõputöö raames ei testitud, kui kaua värav kauem töötaks, kui kolme sensori asemel oleks üks sensor.

Müügikõlblikkuse saavutamiseks peab veel panustama kompaktsusesse ja disaini, sest süsteem peab olema kerge ja mugav paigaldada. Lisaks peab süsteemi olema lihtne kaasas kanda. See annaks suure eelise konkurentide seas, sest enamikel koosneb süsteem mitmest suurest ja raskest kohvrist.

9 Kokkuvõte

Lõputöö eesmärk oli luua juhtmevaba harrastajatele koerte takistusvõistluse treeninguteks ajavõtusüsteem. Erinevalt olemasolevatest lahendustest pidi antud lõputöö lahendus olema odavam, kergem paigaldada ja võimaldama üksi koertega treenida. Tehtav lahendus pidi kasutama laialt levinud elektroonikakomponente ja olema hallatav harrastaja nutitelefonist.

Töö analüüs andis ülevaate planeeritud funktsionaalsustest ja võimalikest tehnoloogiatest kui ka platvormidest. Analüüs aitas kiirendada arendusprotsessi ja andis parimad vahendid antud probleemi lahendamiseks. Lõputöö praktilist teostust kirjeldav peatükk andis ülevaate süsteemi arhitektuurist, arendusprotsessist nii riistvaraliselt kui ka tarkvaralisest aspektist. Testimine on lahutamatu osa arendusest ja aitas välja selgitada enne teadmatuid probleeme. Edasiarendused annavad hea ülevaate, kuhu poole projekt edasi tüürib.

Lõputöö alguses püstitatud eesmärk leidis osaliselt lahenduse. Ajavõtusüsteemi ehitamise mahust ja keerukusest tingitud tagasilöögid ei võimaldanud müügikõlbulikku toodet antud lõputöö raames täielikult valmis saada. Maailma hetkel lukus hoidev koroonaoht ei aidanud sellele kaasa ja süsteemi mitmed katsetamised treeningutel jäid ära. Prototüüp on kinnitatud ja kõik välja toodud nõuded on teostatavad. Klientrakendus vajab veel tugevat lihvi, et see harrastajale mugav ja kasutuskõlblik oleks. Antud süsteemi ei visata nurka, vaid arendatakse edasi kuni kõik eesmärgid on teostatud.

Kasutatud kirjandus

[1] Clean run, *History of Agility, Part 1*, 2004 [Online]. Loetud aadressil: https://www.cleanrun.com/feature/history_of_agility_part_1/index.cfm Kasutatud: 22.03.2021

[2] Kristi Vaidla. *Agility ja mina*, 2012 [Online]. Loetud aadressil: https://www.lexberrys.eu/?page_id=30 Kasutatud: 22.03.2021

[3] STMicroelectronics, *P-NUCLEO-WB55*, 2019 [Online]. Loetud aadressil: https://www.st.com/resource/en/data_brief/p-nucleo-wb55.pdf Kasutatud: 06.04.2021

[4] Espressif Systems (Shanghai) CO., LTD, *ESP32*, 2021 [Online]. Loetud aadressil: https://www.espressif.com/en/products/socs/esp32 Kasutatud: 06.04.2021

[5] Espressif Systems (Shanghai) CO., LTD , *ESP8266*, 2021 [Online] Loetud aadressil: https://www.espressif.com/en/products/socs/esp8266 Kasutatud: 06.04.2021

[6] Arduino, *Arduino UNO WiFi Rev.2*, 2021 [Online] Loetud aadressil: https://store.arduino.cc/arduino-uno-wifi-rev2 Kasutatud: 06.04.2021

[7] Arduino, *Arduino Uno Rev3*, 2021 [Online] Loetud aadressil: https://store.arduino.cc/arduino-uno-rev3 Kasutatud: 06.04.2021

[8] Shenzhen Xunlong Software CO.,Limited, *Orange Pi 4B*, 2021 [Online] Loetud aadressil: http://www.orangepi.org/Orange%20Pi%204B/ Kasutatud: 06.04.2021

[9] Shenzhen Xunlong Software CO.,Limited, *Orange Pi 3*, 2021 [Online] Loetud aadressil: http://www.orangepi.org/Orange%20Pi%203/ Kasutatud: 06.04.2021

[10] Baeldung, *Spring Framework Introduction*, 2020 [Online] Loetud aadressil: https://www.baeldung.com/spring-intro Kasutatud: 08.04.2021

[11] Back4App, *Top 10 backend frameworks*, 2021 [Online] Loetud aadressil: https://blog.back4app.com/backend-frameworks/ Kasutatud: 08.04.2021

[12] J. Neitz, T. Geist ja G.H. Jacobs, *Color vision in the dog*. USA: Cambridge University Press, 1989

[13] Omron Corporation, *E3FA, E3RA, E3FB, E3RB*, 2014 [Online] Loetud aadressil: http://www.ia.omron.com/products/family/3130/specification.html Kasutatud: 19.04.2021

[14] Microsoft, *Introduction to ASP.NET Core SignalR*, 2019 [Online] Loetud aadressil: https://docs.microsoft.com/en-us/aspnet/core/signalr/introduction?view=aspnetcore-5.0 Kasutatud: 19.04.2021
[15] Github, *SignalR Transport Protocols*, 2019 [Online] Loetud aadressil: https://github.com/dotnet/aspnetcore/blob/main/src/SignalR/docs/specs/TransportProtocols.md Kasutatud: 19.04.2021
[16] Github, *Performance tests between REST, gRPC-Web and SignalR from browser*, 2020 [Online] Loetud aadressil: https://github.com/vbilopav/WsBenchmarks/blob/master/readme.md Kasutatud: 19.04.2021
[17] Tallinna Tehnikaülikooli IT Kolledž, *Trükiplaadi valmistamine*, 2014 [Online] Loetud aadressil: https://wiki.itcollege.ee/index.php/Tr%C3%BCkiplaadi_valmistamine Kasutatud: 21.04.2021
[18] Shenzhen Xunlong Software CO.,Limited, *Orange Pi PC Plus*, 2021 [Online] Loetud aadressil: http://www.orangepi.org/orangepipcplus/ Kasutatud: 06.04.2021
[19] Sinovoip, *BPI-M4*, 2021 [Online] Loetud aadressil: http://www.banana-pi.org/m4.html Kasutatud: 06.04.2021
[20] Sinovoip, *BPI-M5*, 2021 [Online] Loetud aadressil: http://www.banana-pi.org/m5.html Kasutatud: 06.04.2021
[21] Hackr.io, *10 Best Web Development Frameworks*, 2020 [Online] Loetud aadressil: https://hackr.io/blog/web-development-frameworks Kasutatud: 06.04.2021
[22] DogStar Kennel, *Fancy Pet Rat Agility*, 2021 [Online] Loetud aadressil: https://www.theagilerat.com/ Kasutatud: 21.04.2021
[23] DogStar Kennel, *Getting Started With Rabbit Agility*, 2021 [Online] https://rabbitagility.com/?id=61 Kasutatud: 21.04.2021
[24] Federation Cynologique Internationale, *Agility*, 2020 [Online] Loetud aadressil: http://www.fci.be/en/Agility-45.html Kasutatud: 07.04.2021
[25] Galican, *Timers*, 2021 [Online] Loetud aadressil: https://www.galican.com/en/69-timers Kasutatud: 09.04.2021
[26] Tessa-Timer, *Price*, 2021 [Online] Loetud aadressil: https://www.tessa-timer.eu/index.php/price Kasutatud: 09.04.2021
[27] Bravedog SIA, BRAVEDOG Agility Timing System, 2021 [Online] Loetud aadressil: http://bravedog.lv/gb/content/6-agility-timing-system Kasutatud: 09.04.2021

[28] Dog agility timer, *Barky V3*, 2021 [Online] Loetud aadressil: https://barkr.eu/ Kasutatud: 09.04.2021
[29] *Cards and security devices for personal identification — Contactless vicinity objects — Part 3: Anticollision and transmission protocol*, ISO/IEC 15693-3:2019. [Online] Loetud aadressil: https://www.iso.org/standard/73602.html Kasutatud: 10.04.2021
[30] *HDTV Formats Technical Report*. Geneva: EBU TECHNICAL. 13. February 2010.
[31] Process of assembling electrical circuits, by Abramson Moe, Stanislaus F. Danko (1950, Aug. 28). US2756485A. [Online]. Loetud aadressil: https://patents.google.com/patent/US2756485A/en
[32] Espressif Systems (Shanghai) CO., LTD , *ESP32-S2*, 2021 [Online] https://www.espressif.com/en/products/socs/esp32-s2 Kasutatud: 06.04.2021
[34] Red Hat, Inc., *What is Docker?*, 2021 [Online] Loetud aadressil: https://opensource.com/resources/what-docker Kasutatud: 10.04.2021
[35] Docker, Inc., *Dockeri koduleht*, 2021 [Online] Loetud aadressil: https://www.docker.com/ Kasutatud: 10.04.2021
[36] JEDEC, *Embedded Multi-Media Card (e•MMC), Electrical Standard (5.1)*, 2019 [Online] Loetud aadressil: https://www.jedec.org/standards-documents/docs/jesd84-b51 Kasutatud: 10.04.2021
[37] Raspberry Pi Foundation, *Raspberry Pi 4 Tech Specs*, 2021 [Online] Loetud aadressil: https://www.raspberrypi.org/products/raspberry-pi-4-model-b/specifications/ Kasutatud: 06.04.2021
[38] Ruby Community, *About Ruby*, 2021 [Online]. https://www.ruby-lang.org/en/about/ Kasutatud: 11.04.2021
[39] Laravel LLC, *Laravel Introduction*, 2021 [Online] Loetud aadressil: https://laravel.com/docs/8.x Kasutatud: 11.04.2021
[40] Django Software Foundation, *Django Overview*, 2021 [Online] Loetud aadressil: https://www.djangoproject.com/start/overview/ Kasutatud: 11.04.2021
[41] OpenJS Foundation, *Express.js koduleht*, 2021 [Online] Loetud aadressil: https://expressjs.com/ Kasutatud: 11.04.2021
[42] Microsoft, *What is ASP.NET Core?*, 2021 [Online] Loetud aadressil: https://dotnet.microsoft.com/learn/aspnet/what-is-aspnet-core Kasutatud: 11.04.2021

[43] Android, *What is Android*, 2021 [Online] Loetud aadressil: https://www.android.com/what-is-android/ Kasutatud: 11.04.2021
[44] Recombu, *What is iOS and what does iOS stand for?*, 2020 [Online] Loetud aadressil: https://recombu.com/mobile/article/what-is-ios-and-what-does-ios-stand-for Kasutatud: 12.04.2021
[45] Kotlin Foundation, *Kotlini koduleht*, 2021 [Online] Loetud aadressil: https://kotlinlang.org/ Kasutatud: 12.04.2021
[46] Apple Inc, *Swift*, 2021 [Online] Loetud aadressil: https://developer.apple.com/swift/ Kasutatud: 12.04.2021
[47] Google, *Flutter*, 2021 [Online] Loetud aadressil: https://flutter.dev/ Kasutatud: 12.04.2021
[48] Google, *Dart*, 2021 [Online] Loetud aadressil: https://dart.dev/ Kasutatud: 12.04.2021
[49] Google, *Flutter performance profiling*, 2021 [Online] Loetud aadressil: https://flutter.dev/docs/perf/rendering/ui-performance Kasutatud: 12.04.2021
[50] Microsoft, *What is Xamarin?*, 2021 [Online] Loetud aadressil: https://dotnet.microsoft.com/learn/xamarin/what-is-xamarin Kasutatud: 12.04.2021
[51] Microsoft, *An Introduction to NuGet*, 2019 [Online] Loetud aadressil: https://docs.microsoft.com/en-us/nuget/what-is-nuget Kasutatud: 12.04.2021
[52] Facebook, Inc., *React Native koduleht*, 2021 [Online] Loetud aadressil: https://reactnative.dev/ Kasutatud: 12.04.2021
[53] Mozilla, What is JavaScript?, 2021 [Online] Loetud aadressil: https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/What_is_JavaScript Kasutatud: 13.04.2021
[54] Mozilla, *HTML Basics*, 2021 [Online] Loetud aadressil: https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/HTML_basics Kasutatud: 13.04.2021
[55] Mozilla, What is CSS?, 2021 [Online] Loetud aadressil: https://developer.mozilla.org/en-US/docs/Learn/CSS/First_steps/What_is_CSS Kasutatud: 13.04.2021
[56] Guru99, What is C Programming Language? Basics, Introduction, History, 2021 [Online] Loetud aadressil: https://www.guru99.com/c-programming-language.html Kasutatud: 13.04.2021

[57] Arduino, What is Arduino?, 2021 [Online] Loetud aadressil: https://www.arduino.cc/en/guide/introduction Kasutatud: 13.04.2021
[58] Espressif Systems (Shanghai) CO., LTD, *ESP8266_RTOS_SDK Get started*, 2021 [Online] Loetud aadressil: https://docs.espressif.com/projects/esp8266-rtos-sdk/en/latest/get-started/index.html Kasutatud: 13.04.2021
[59] Espressif Systems (Shanghai) CO., LTD, *ESP_IDF Get started*, 2021 [Online] Loetud aadressil: https://docs.espressif.com/projects/esp-idf/en/latest/esp32/get-started/ Kasutatud: 13.04.2021
[60] STMicroelectronics, *STM32CubeIDE: The First Free ST IDE with STM32CubeMX Built-in*, 2019 [Online] Loetud aadressil: https://blog.st.com/stm32cubeide-free-ide/ Kasutatud: 14.04.2021
[61] Medium, *What is an ORM and Why You Should Use it*, 2018 [Online] Loetud aadressil: https://blog.bitsrc.io/what-is-an-orm-and-why-you-should-use-it-b2b6f75f5e2a Kasutatud: 14.04.2021
[62] Facebook, Inc., *React.js Getting Started*, 2021 [Online] Loetud aadressil: https://reactjs.org/docs/getting-started.html Kasutatud: 14.04.2021
[63] Mozilla, *Introduction to the DOM*, 2021 [Online] Loetud aadressil: https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model/Introduction Kasutatud: 15.04.2021
[64] Facebook, Inc., *Introducing JSX*, 2021 [Online] Loetud aadressil: https://reactjs.org/docs/introducing-jsx.html Kasutatud: 15.04.2021
[65] Refsnes Data, Introduction to XML, 2021 [Online] Loetud aadressil: https://www.w3schools.com/xml/xml_whatis.asp Kasutatud: 15.04.2021
[66] Google, *What is Angular?*, 2021 [Online] Loetud aadressil: https://angular.io/guide/what-is-angular Kasutatud: 15.04.2021
[67] Vue, *What is Vue?*, 2021 [Online] Loetud aadressil: https://vuejs.org/v2/guide/ Kasutatud: 16.04.2021
[68] Refsnes Data, jQuery Introduction, 2021 [Online] Loetud aadressil: https://www.w3schools.com/jquery/jquery_intro.asp Kasutatud: 16.04.2021
[69] Blue Spire Inc., What is Aurelia?, 2021 [Online] Loetud aadressil: https://aurelia.io/docs/overview/what-is-aurelia/ Kasutatud: 16.04.2021
[70] Titanium I.T. LLC, *James Shore: Dependency Injection Demystified,* 2006, [Online]. Loetud aadressil:

http://www.jamesshore.com/v2/blog/2006/dependency-injection-demystified Kasutatud: 16.04.2021
[71] Microsoft, *Typescripti koduleht*, 2021 [Online] Loetud aadressil: https://www.typescriptlang.org/ Kasutatud: 16.04.2021
[72] Trygve Reenskaug, *THING-MODEL-VIEW-EDITOR - an Example from a planningsystem*. Xerox PARC, May 1979.
[73] Educative, Inc., *MVC Architecture in 5 minutes: a tutorial for beginners*, 2020 [Online] Loetud aadressil: https://www.educative.io/blog/mvc-tutorial Kasutatud: 17.04.2021
[74] HiveMQ GmbH, *Introducing the MQTT Protocol - MQTT Essentials: Part 1*, 2015 [Online] Loetud aadressil: https://www.hivemq.com/blog/mqtt-essentials-part-1-introducing-mqtt/ Kasutatud 17.04.2021
[75] HiveMQ GmbH, *Publish & Subscribe - MQTT Essentials: Part 2*, 2015 [Online] Loetud aadressil: https://www.hivemq.com/blog/mqtt-essentials-part2-publish-subscribe/ Kasutatud 17.04.2021
[76] *Transmission Control Protocol*, RFC 759. [Online] Loetud aadressil: https://tools.ietf.org/html/std7 Kasutatud: 18.04.2021
[77] *The Transport Layer Security (TLS) Protocol Version 1.3*, RFC 8446. [Online] Loetud aadressil: https://tools.ietf.org/html/rfc8446 Kasutatud: 18.04.2021
[78] *The Secure Sockets Layer (SSL) Protocol Version 3.0*, RFC 6101. [Online] Loetud aadressil: https://tools.ietf.org/html/rfc6101 Kasutatud: 18.04.2021
[79] HiveMQ GmbH, *Quality of Service 0,1 & 2 - MQTT Essentials: Part 6*, 2015 [Online] Loetud aadressil: https://www.hivemq.com/blog/mqtt-essentials-part-6-mqtt-quality-of-service-levels/ Kasutatud 17.04.2021
[80] *Hypertext Transfer Protocol -- HTTP/1.1*, RFC 2616. [Online] Loetud aadressil: https://tools.ietf.org/html/rfc2616 Kasutatud: 18.04.2021
[81] *The Constrained Application Protocol (CoAP)*, RFC 7252. [Online] Loetud aadressil: https://tools.ietf.org/html/rfc7252 Kasutatud: 18.04.2021
[82] *Guidelines and Registration Procedures for URI Schemes*, RFC 7595. [Online] Loetud aadressil: https://tools.ietf.org/html/rfc7595 Kasutatud: 18.04.2021
[83] *User Datagram Protocol*, RFC 768. [Online] Loetud aadressil: https://tools.ietf.org/html/rfc768 Kasutatud: 18.04.2021
[84] *Datagram Transport Layer Security Version 1.2*, RFC 6347. [Online] Loetud aadressil: https://tools.ietf.org/html/rfc6347 Kasutatud: 18.04.2021

[85] OASIS Advanced Message Queuing Protocol (AMQP) Version 1.0 Part 3: Messaging. 29 October 2012. OASIS Standard. [Online] Loetud aadressil: http://docs.oasis-open.org/amqp/core/v1.0/amqp-core-messaging-v1.0.html Kasutatud: 19.04.2021

[86] *Information technology — Open Systems Interconnection — Basic Reference Model: The Basic Model*, ISO/IEC 7498-1:1994. [Online] Loetud aadressil: https://www.iso.org/standard/20269.html Kasutatud: 18.04.2021

[87] D. Thangavel, X. Ma, A. Valera, H.-X. Tan, ja C. K.-Y. Tan, *Performance evaluation of MQTT and CoAP via a common middleware*, in Intelligent Sensors, Sensor Networks and Information Processing, 2014 IEEE Ninth International Conference on. IEEE, 2014, pp. 1–6

[88] S. Bandyopadhyay and A. Bhattacharyya, *Lightweight internet protocols for web enablement of sensors using constrained gateway devices*, in Computing, Networking and Communications (ICNC), 2013 International Conference on. IEEE, 2013, pp. 334–340

[89] A. Foster, *Messaging technologies for the industrial internet and the internet of things whitepaper*, PrismTech, 2015.

[90] R. S. Cohn, *A comparison of AMQP and MQTT*, StormMQ, 2011.

[91] Mozilla, *What's AJAX?*, 2021 [Online] Loetud aadressil: https://developer.mozilla.org/en-US/docs/Web/Guide/AJAX/Getting_Started Kasutatud: 21.04.2021

[92] Microsoft, *Introduction to SignalR*, 2014 [Online] Loetud aadressil: https://docs.microsoft.com/en-us/aspnet/signalr/overview/getting-started/introduction-to-signalr Kasutatud: 21.04.2021

[93] gRPC, *Introduction to gRPC*, 2020 [Online] https://grpc.io/docs/what-is-grpc/introduction/ Kasutatud 21.04.2021

[94] Arrow Electorincs, Inc., *What is IMU? Inertial Measurement Unit Working & Applications*, 2018 [Online] Loetud aadressil: https://www.arrow.com/en/research-and-events/articles/imu-principles-and-applications Kasutatud: 21.04.2021

[95] Eesti Kennelliit, *FCI Agility World Championship 2021 Official announcement*, 2021 [Online] Loetud aadressil: https://www.facebook.com/agility2021/posts/814273669513326 Kasutatud: 12.05.2021

Lisa 1 – Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks[1]

Mina, Kerman Saapar

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose „Individuaalne ajavõtusüsteem koerte takistusvõistluse treeninguteks“, mille juhendaja on Madis Listak ja kaasjuhendaja on Meelis Antoi.
   1.1. reprodutseerimiseks lõputöö säilitamise ja elektroonse avaldamise eesmärgil, sh Tallinna Tehnikaülikooli raamatukogu digikogusse lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;
   1.2. üldsusele kättesaadavaks tegemiseks Tallinna Tehnikaülikooli veebikeskkonna kaudu, sealhulgas Tallinna Tehnikaülikooli raamatukogu digikogu kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.
2. Olen teadlik, et käesoleva lihtlitsentsi punktis 1 nimetatud õigused jäävad alles ka autorile.
3. Kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest ning muudest õigusaktidest tulenevaid õigusi.

28.04.2021

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