TALLINNA TEHNIKAÜLIKOOL
INSENERITEADUSKOND
Virumaa kolledž

Lifti vintsi mootorite rikete jälgimine

Fault monitoring of elevator winch motors

ARUKAD SÜSTEEMID JA RAKENDUSINFOTEHNOLOOGIA ÕPPEKAVA
LÕPUTÖÖ

Üliõpilane: Aleksandr Lendi
Üliõpilaskood: 207569EDTR
Juhendaja: Sergei Pavlov, lektor

Kohtla-Järve, 2024

AUTORIDEKLARATSIOON

Olen koostanud lõputöö iseseisvalt.

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

"...." ..... 20.....

Autor: .....
/ allkiri /

Töö vastab rakenduskõrgharidusõppe lõputööle/magistritööle esitatud nõuetele
"...." ..... 20.....

Juhendaja: .....
/ allkiri /

Kaitsmisele lubatud
"...." ..... 20.....

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

LIHTLITSENTS LÕPUTÖÖ ÜLDSUSELE KÄTTESAADAVAKS TEGEMISEKS JA REPRODUTSEERIMISEKS

Mina Aleksandr Lendi (sünnikuupäev:05.05.2000)

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

lifti vintsi mootorite rikete jälgimine, mille juhendaja on Sergei Pavlov,

1.1. reprodutseerimiseks säilitamise ja elektroonilise avaldamise eesmärgil, sealhulgas 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 punktis 1 nimetatud õigused jäävad alles ka autorile.

3. Kinnitan, et lihtlitsentsi andmisega ei rikuta kolmandate isikute intellektuaalomandi ega isikuandmete kaitse seadusest ja teistest õigusaktidest tulenevaid õigusi.

SISUKORD

LÜHENDITE JA TÄHISTE LOETELU 5
SISSEJUHATUS 6
1 MOOTORIDIAGNOSTIKA SÜSTEEMIDE VÕRDLUSANALÜÜS 8
1.1 Mootorite diagnostikasüsteemide võrdlus 8
1.2 Mootoridiagnostika komponentide ja tööriistade valik 11
1.2.1 Kontrolleri valik 11
1.2.2 Helianduri valimine 12
1.2.3 Temperatuurianduri valimine 13
1.2.4 Vibratsioonianduri valimine 14
1.2.5 Andmete salvestamine 14
1.2.6 Andmetöötlus 15
1.2.7 Teadete saatmine 15
1.2.8 Andmete kuvamine 15
2 SÜSTEEMI ARENDUS JA TESTIMINE 17
2.1 Füüsilise prototüübi väljatöötamine ja kokkupanek 17
2.2 Tarkvara arendamine 18
2.2.1 Kontrollerile programmi kirjutamine 18
2.2.2 Serveripoolne arendamine 21
2.2.3 Andmebaasi seadistamine 24
2.2.4 Veebiliidese loomine 27
2.3 Testimine ja silumine 30
2.3.1 Funktsionaalsuse testimine 30
2.3.2 Silumine ja tõrkeotsing 33
2.4 Saadud andmete analüüs 34
KOKKUVÕTE 37
SUMMARY 38
KASUTATUD KIRJANDUSE LOETELU 40

LÜHENDITE JA TÄHISTE LOETELU

API (Application Programming Interface) - Määratluste ja protokollide komplekt tarkvara arendamiseks ja integreerimiseks. API võimaldab erinevatel rakendustel üksteisega suhelda, ilma et oleks vaja teada nende rakendamise üksikasju.

Arduino - Avatud platvorm elektrooniliste seadmete prototüüpide loomiseks, kasutades paindlikku, hõlpsasti kasutatavat riist- ja tarkvara. Arduinot kasutatakse kontrollerina andurite andmete kogumiseks ja töötlemiseks.

Flask - Mikroraamistik Pythoni programmeerimiskeeles veebirakenduste loomiseks. Flaski kasutatakse seiresüsteemi serveriosa arendamiseks.

GPIO (General-Purpose Input/Output) - Üldotstarbeline sisend/väljund, liides mikrokontrollerite liidestamiseks teiste seadmete või keskkonnaga.

HTTPS (HyperText Transfer Protocol Secure) - HTTP-protokolli laiendus, mis toetab andmeedastuse turvalisuse parandamiseks krüptimist.

JSON (JavaScript Object Notation) - Tekstipõhine andmevahetusvorming, mida on lihtne lugeda ja kirjutada nii inimestele kui ka masinate poolt analüüsimiseks ja genereerimiseks. Kasutatakse andmete edastamiseks serveri ja veebikliendi vahel.

JSX (JavaScript XML) - JavaScripti süntaksilaiend, mida kasutatakse rakenduses React, et kirjeldada rakenduse liidese struktuuri, kasutades HTML-i sarnast koodi. JSX pakub mugavat viisi Reacti elementide ja komponentide loomiseks, võimaldades arendajatel visuaalselt kirjeldada kasutajaliidese struktuuri koodis, mis seejärel teisendatakse JavaScripti objektideks. JSX-i kasutamine Reactis vähendab arendusaega ja lihtsustab dünaamiliste veebirakenduste loomise protsessi, pakkudes kasutajaliidese komponentide visuaalsemat esitust.

PostgreSQL - Võimas avatud andmebaasihaldussüsteem. Kasutatakse seiresüsteemi kogutud andmete salvestamiseks ja töötlemiseks.

React.js - JavaScripti teek kasutajaliideste arendamiseks. Kasutatakse seiresüsteemi veebiliidese loomiseks.

SISSEJUHATUS

Tänapäeva kiires linnakeskkonnas, kus kõrged hooned ja elutempo nõuavad tõhusat vertikaalset transporti, muutuvad liftid üha olulisemaks. Liftid ei tee mitte ainult igapäevaelu lihtsamaks, vaid on muutunud ka kaasaegsete hoonete südameks, tagades nende funktsionaalsuse. Seetõttu tekkis paratamatult küsimus liftide töökindluse ja ohutuse kohta. Liftisüsteemide vananemine ja kõrgemate ohutusstandardite kasutuselevõtt on toonud kaasa vajaduse tõhusama diagnostikasüsteemi järele, eriti vanemate liftide puhul, mille disain ja tehnoloogia ei vasta sageli tänapäevastele nõuetele.

Antud lõputöö hõlmab vanade liftimootorite diagnoosimise ja jälgimise süsteemi väljatöötamist, mis on oluline linnataristu arengu ja liftide vananemise kontekstis. Arvestades liftide kasvavat arvu ja nende pikemat kasutusiga, on vanemate liftisüsteemide töökindluse ja ohutuse parandamise vajadus kriitilise tähtsusega. Tehnoloogiliste lahenduste, näiteks Arduino-põhiste süsteemide juurutamine võimaldab mitte ainult tõhusalt jälgida seadmete seisukorda, vaid ka kiiresti reageerida võimalikele tõrgetele nende töös.

Lõputöö käigus on plaanis luua kaasaegne ja töökindel diagnostikasüsteem, mis on hõlpsasti integreeritav olemasolevasse liftisüsteemi ning töötab tõhusalt õnnetuste ärahoidmiseks ja hoolduskulude vähendamiseks. Uuringu olulisus tuleneb otseselt vajadusest parandada liftide kasutamise ohutust ja seeläbi vähendada võimalike õnnetuste ohtu. Oluline on ka majanduslik aspekt - süsteemi efektiivsus aitab vähendada liftide hooldusele ja remondile kuluvat summat. Uuringus käsitletakse Arduino platvormi kasutamise eeliseid diagnostiliste protseduuride lihtsustamiseks, pakkudes samal ajal paindlikkust ja kohanemisvõimet.

Metoodiline baas sisaldab olemasolevate diagnostikasüsteemide võrdlevat analüüsi, sobivate andurite ja nende konfiguratsiooni valikut. Samuti töötatakse välja spetsiaalne tarkvara ning luuakse Arduino baasil prototüüp, mis võimaldab reaalajas andmeid koguda ja analüüsida. Selle süsteem peaks tagama diagnostikaprotsessi suurema täpsuse ja kiiruse, andes hooldusmeeskondadele võimaluse kiiresti reageerida mis tahes tuvastatud probleemidele.

Selle lõputöö ootused on seotud liftimootorite diagnostika, seiresüsteemi loomise, ohutuse ja kasutusmugavusega. Lõputöö eesmärk on luua odav ja tõhus süsteem, mis mitte ainult ei suuda tuvastada rikkeid varajases staadiumis, vaid annab ka hooldusmeeskonnale kiiret ja täpset teavet rikke olemuse kohta. See võimaldab neil teha teadlikke otsuseid ja võtta asjakohaseid meetmeid, et tagada liftide ohutu ja usaldusväärne töö.

Lõputöö ülesanded on:

1. Liftimootorite diagnostikasüsteemi väljatöötamine, sh andurite valik, andmete kogumine ja töötlemine.

2. Füüsilise prototüübi loomine ja süsteemi testimine, et hinnata selle praktilist rakendatavust ja tõhusust.

3. Töökindluse ja ohutuse tagamiseks vajalike tehnoloogiliste lahenduste väljatöötamine.

Lõppkokkuvõttes loodetakse, et lõputöö mitte ainult ei paranda vanemate liftide ohutust ja töökindlust, vaid loob ka aluse edasistele uuendustele tööstuses, tagades, et liftisüsteemid saavad areneda nii, et need vastaksid kaasaegse linnataristu nõuetele.

Võtmesõnad: liftimootorite diagnostika, Arduino, reaalajas monitooring, liftide ohutus, bakalaureusetöö.

1 MOOTORIDIAGNOSTIKA SÜSTEEMIDE VÕRDLUSANALÜÜS

Vanade liftide elektrimootorite diagnostika- ja juhtimissüsteemi juurutamiseks otsustati luua mikrokontrolleril põhine süsteem. See võimaldab anduritelt pidevalt andmeid vastu võtta, neid töödelda ja vajadusel edasiseks analüüsiks serverisse saata.

Lõputöö nõuete analüüsi põhjal otsustati kasutada kolme tüüpi andureid: temperatuuri-, vibratsiooni- ja heliandurid. Uuringute kohaselt viitavad vibratsiooninäidud sageli mootori mehaanilistele probleemidele, näiteks kulunud laagritele, mis võib mootori töötamise ajal põhjustada vibratsiooni suurenemist [1]. Lisaks võib mootori töö ajal esinevate helimustrite anomaaliate tuvastamine viidata erinevatele riketele, mida saab tuvastada kiireks reageerimiseks ja tõsiste töötõrgete ennetamiseks [2].

Temperatuuri jälgimise tähtsus on oluline ka seetõttu, et kõrvalekalded temperatuurinäitudes võivad viidata ülekuumenemisele või muudele kriitilistele muutustele, mis nõuavad kohest tähelepanu. Temperatuuriandur suudab selliseid tingimusi varakult tuvastada, mis on oluline seadme ohutu töö tagamiseks. Temperatuuri, müra ja vibratsiooni taset ühendav seiresüsteem aitab luua terviklikuma pildi lifti mootori seisukorrast, mis on tänapäevaste diagnostikameetodite juures oluline [3].

Üldiselt võimaldab nende andurite kasutamine kiiresti tuvastada ja analüüsida lifti mootori jõudluse erinevaid aspekte, parandades ohutust ja tõhusust.

1.1 Mootorite diagnostikasüsteemide võrdlus

Analüüsi eesmärgil võrreldakse mitmeid mootoridiagnostika süsteeme, milleks on Arduino R4 WiFi [4], Omron K6CM-CI2M [5], Schneider Electric PowerLogic ION9000 [6] ja ABB Ability™ Smart Sensor [7], [8]. Iga süsteemi hinnatakse mitme kriteeriumi alusel, sealhulgas andurite tüübid, diagnostikafunktsioonid, kaugseire võimalused, integratsioon olemasoleva infrastruktuuriga, töökindlus ja täpsus ning maksumus. Saadud andmed annavad põhjaliku võrdluse iga süsteemi tõhususe ja kasutatavuse kohta mootori diagnostikas.

Vanade elektrimootorite erinevate diagnostikameetodite võrdlemiseks kasutati võrdlustabeli andmeid (vt tabel 1). Teavet on saadud erinevatest allikatest, sealhulgas tootjate ametlikelt veebisaitidelt ja toote tehnilistest kirjeldustest.

Tabel 1. Motoorse diagnostika lahenduste võrdlus

Kriteeriumid	**Arduino R4 WiFi (temperatuuri-, heli- ja vibratsioonidetektoritega)**	**Omron K6CM-CI2M**	**Schneider Electric PowerLogic ION9000**	**ABB Ability™ Smart Sensor**
Andurite tüübid	Temperatuur, vibratsioon, heli	Vool, pinge, sagedus	Vool, pinge, sagedus, võimsus, harmooniad	Temperatuur, vibratsioon, muud mootori tööparameetrid
Diagnostikafunktsionaalsus	Tarkvara andmete analüüsimiseks	Sisseehitatud diagnostikaalgoritm	Anomaaliate tuvastamine elektrienergias, andmete analüüs	Mootori töö jälgimine ebanormaalsete tingimuste, kulumise ja rikete suhtes
Kaugseire võimalused	WiFi, andmete saatmine serverisse	Modbus TCP/IP, Ethernet/IP	Juhtmevaba side andmete edastamiseks pilveteenustesse	Juhtmevaba side andmete edastamiseks pilveteenustesse
Integreerimine olemasolevasse infrastruktuuri	Kohanemine erinevate mootorimudelitega	Liftikompleksside ühilduvus	Lihtne integreerimine erinevate mootoritüüpide ja juhtimissüsteemidega	Lihtne integreerimine erinevate mootoritüüpide ja juhtimissüsteemidega
Usaldusväärsus ja täpsus	Sõltub tarkvarast ja andurite kvaliteedist	Kõrge	Kõrge mõõtmiste täpsus ja usaldusväärsus	Usaldusväärne ja täpne diagnostika andmeanalüüsi algoritmidega

Kriteeriumid	**Arduino R4 WiFi (temperatuuri-, heli- ja vibratsioonidetektoritega)**	**Omron K6CM-CI2M**	**Schneider Electric PowerLogic ION9000**	**ABB Ability™ Smart Sensor**
Maksumus	Suhteliselt madal, sõltub andurite ja komponentide valikust	Keskmine	Väga kõrge	Hind on saadaval ainult klientidele päringu alusel

Tabeliandmete põhjal saab teha järgmised järeldused:

1. Arduino R4 WiFi diagnostikameetodil pole mitte ainult lai valik andureid ja funktsioone, vaid ka madal hind, mis muudab selle korteriühistutele atraktiivseks.
2. Seade Omron K6CM-CI2M tagab kõrge diagnostilise täpsuse, kuid selle maksumus võib olla paljudele tarbijatele taskukohane.
3. Schneider Electric PowerLogic ION9000 ja ABB Ability™ Smart Sensor, kuigi väga töökindlad ja täpsed, muudab nende maksumus nende paigaldamiseks elamuliftisüsteemidesse ebapraktiliseks.

Järeldus: Professionaalsed mootoridiagnostika süsteemid, nagu Schneider Electric PowerLogic ION9000 ja ABB Ability™ Smart Sensor, tagavad suure täpsuse ja töökindluse, kuid nende maksumus võib olla korteriühistutele üle jõu käiv. Arduino R4 WiFi-põhine diagnostika pakub taskukohase ja hõlpsasti paigaldatava lahenduse.

1.2 Mootoridiagnostika komponentide ja tööriistade valik

See osa keskendub mootori diagnostikasüsteemi komponentide valimise teoreetilisele alusele.

1.2.1 Kontrolleri valik

Võrreldavate kontrollerite hulka kuuluvad Arduino Uno R4 WiFi, Arduino Uno R3 [9] ja Raspberry Pi Pico WH [10], [11]. Allpool on põhiomaduste võrdlustabel:

Tabel 2. Mikrokontrolleri omaduste võrdlus lõputöö jaoks

Kriteerium	**Arduino Uno R4 WiFi**	**Arduino Uno R3**	**Raspberry Pi Pico WH**
Maksumus (€)	€25,00	€24,00	€9,26
WiFi	Sisseehitatud WiFi moodul	Puudub WiFi	Sisseehitatud WiFi moodul
Protsessor	Renesas RA4M1 (Arm Cortex-M4)	ATmega328P	Mikrokontroller RP2040
Mälu	256 kB Flash, 32 kB RAM	32 KB Flash, 2 KB SRAM	264 KB Flash, 26 KB RAM
GPIO	14 digitaalset pini, 6 analoogsisendit	14 digitaalset pini, 6 analoogsisendit	26 GPIO pini
Suurus	Laius 68.85 mm, Pikkus 53.34 mm	Pikkus 68.6 mm, Laius 53.4 mm	Mõõtmed 51 mm x 21 mm x 0.3 mm
Liidesed	USB-C, UART, I2C, SPI, CAN	USB, UART, I2C, SPI, GPIO	USB, UART, I2C, SPI, GPIO
Programmeerimine	C/C++	C/C++	C/C++, CircuitPython, Micropython

Pärast esitatud andmete analüüsi tundub Arduino Uno R4 WiFi selle lõputöö jaoks eelistatavam variant. See valik on tingitud mitmest tegurist. Esiteks sisseehitatud Wi-Fi mooduli olemasolu, mis muudab kontrolleri paindlikumaks ja mugavamaks traadita sidesüsteemide loomiseks. Erinevalt Arduino Uno R3-st, kus Wi-Fi moodul tuleb eraldi osta. Lisaks võimaldab analoogsisendite olemasolu lugeda anduritelt analoognäite, mis on oluline elektrimootori diagnostikasüsteemi jaoks.

Lisaks on Arduino Unol ulatuslik andmebaas diagnostikaseadmete loomise kohta Internetis, mis lihtsustab arendamist ja juurutamist. Autoril on Arduino Uno kontrolleritega töötamise kogemus juba olemas, mis mängib samuti rolli antud kontrolleri kasuks otsustamisel.

Järeldus: kuigi Raspberry Pi Pico WH on soodsam variant, eelistas autor Arduino Uno R4 WiFit selle mugavuse, paindlikkuse ja laia funktsionaalsuse ning varasema kogemuse arduino kontrolleritega.

1.2.2 Helianduri valimine

Võrdlustabelis on toodud Arduino kolme erineva helianduri omadused: Keyestudio Analog Sound Sensor [12], DFRobot Sound Sensor [13] ja Adafruit Electret Microphone Amplifier [14]. Neid andureid kasutatakse helitugevuse tuvastamiseks keskkonnas ja neil on analoogväljundsignaal.

Tabel 3. Heliandurite võrdlus

Kriteerium	**Keyestudio analoog-helitugevusandur**	**DFRobot helitugevusandur**	**Adafruiti elektreetmikrofoni võimendi**
Tööpinge	3.3V-5V (DC)	3.3V-5V (DC)	2.4V-5V (DC)
Väljundsignaal	Analoogiline	Analoogiline	Analoogiline
Hind (€)	3,42	4,06	6,41

Järeldus: Kõikide vaadeldavate andurite hind on sarnane ja kõik need sobivad oma funktsionaalsuselt antud lõputöö jaoks. Ostuvõimaluse ja pika tarnevajaduse puudumise tõttu valiti aga Keyestudio Analog Sound Sensor.

1.2.3 Temperatuurianduri valimine

Võrdlustabelis on toodud kahe erineva Arduino temperatuurimõõtmisanduri omadused: Shillehtek DHT22 [15], Keyestudio DS18b20 [16]. Neid andureid kasutatakse ümbritseva õhu temperatuuri mõõtmiseks. Anduritel on digitaalne väljundsignaal.

Tabel 4. Temperatuuriandurite võrdlus

Kriteerium	**Shillehtek DHT22**	**Keyestudio DS18b20**
Tööpinge	3.3V-5V (DC)	3.3V-5V (DC)
Temperatuuri mõõtepiirkond	-40°C kuni 80°C	-55°C kuni +125°C
Temperatuuri mõõtmise täpsus	±0.5°C	±0.5°C
Väljundsignaal	Digitaalne	Digitaalne
Niiskustaseme mõõtmine	Jah	Ei
Hind (€)	7,38	6,65

Järeldus: võrdluse põhjal võime järeldada, et Keyestudio DS18b20 temperatuuriandur sobib kõige paremini mootorite diagnoosimiseks. Selle lai temperatuuri mõõtmise vahemik (-55 °C kuni +125 °C) on selle lõputöö jaoks sobivam kui Shillehtek DHT22 (-40 °C kuni 80 °C) ja DHT22 pakutav niiskuse tuvastamise võimalus pole selle konkreetse ülesande jaoks vajalik. Arvestades mõlema anduri peaaegu identset maksumust, tundub, et valik Keyestudio DS18b20 kasuks on kõige õigustatum.

1.2.4 Vibratsioonianduri valimine

Võrdlustabelis on näidatud kahe erineva vibratsioonitaseme anduri omadused Arduino jaoks: Piesoelektriline kilevibratsiooniandur [17], KY-002 vibratsioonilüliti anduri moodul [18]. Neid andureid kasutatakse vibratsioonitaseme mõõtmiseks.

Tabel 5. Vibratsiooniandurite võrdlus

Kriteerium	**Piezoelektriline kilevibratsiooniandur**	**KY-002 vibratsioonilüliti andurimoodul**
Tööpinge	5.0V (DC)	3.3V-5V (DC)
Väljundsignaal	Analoogiline ja digitaalne	Digitaalne (0 või 1)
Tundlikkus	Reguleeritav	Fikseeritud
Mõõtmed	20 x 20 mm	15.5 mm x 20.3 mm
Hind (€)	1,73	6,46

Järeldus: pärast kahe vibratsioonianduri võrdlemist otsustati kasutada piesoelektrilist vibratsiooniandurit. Selle eeliseks on analoogsignaali pakkumine, mis võimaldab täpsemalt mõõta mootori vibratsioonitaset, mis on selle lõputöö jaoks oluline kriteerium. Lisaks ei erine kummagi anduri hind palju, seega ei mängi see anduri valikul määravat rolli.

1.2.5 Andmete salvestamine

Efektiivseks tööks ja andmete salvestamiseks valiti PostgreSQL andmebaas. See valik on tingitud selle töökindlusest, võimsatest võimalustest ja avatud lähtekoodist, mis muudab süsteemi hõlpsaks skaleerimise ja kohandamise lõputöö vajadustele vastavaks.

Andmebaasi majutamiseks valisin teenuse supabase.com, mis pakub PostgreSQL-i andmebaasidele mugavat ja tasuta hosti. See muudab lõputöö jaoks andmebaasi loomise ja seadistamise lihtsaks ning tagab ka kõrge kättesaadavuse ja andmete turvalisuse [19].

Andmebaasi luuakse tabelid andurite teabe salvestamiseks, samuti spetsiaalne tabel normi ületavate andmete jaoks. See võimaldab andmeid tõhusalt jälgida ja analüüsida.

1.2.6 Andmetöötlus

Arduino päringute vastuvõtmiseks ja töötlemiseks otsustati arendada Pythoni programmeerimiskeeles server, kasutades Flask [20] raamistikku. Flask on kerge ja võimas tööriist Pythonis veebirakenduste loomiseks.

Python valiti taustaserveri kirjutamiseks selle lihtsuse ja paindlikkuse, samuti suure arendajate kogukonna ja rikkaliku tööriistade ökosüsteemi tõttu. Python pakub laialdasi võimalusi andmetöötluseks ja andmebaasidega suhtlemiseks, mistõttu on see suurepärane valik andurite andmete töötlemise serveri juurutamiseks.

Flask pakub mugavaid tööriistu veebirakenduste loomiseks, võimaldades kiiresti ja lihtsalt luua API-sid Arduino ja teiste seadmetega suhtlemiseks. See pakub lihtsat ja intuitiivset liidest HTTP-päringute töötlemiseks ja vastuste genereerimiseks, samuti mugavaid tööriistu Pythoni andmebaasidega suhtlemiseks [20].

Flask serveri host kasutab platvormi replit.com. See valik on tingitud väikese projekti tasuta serverihostimise võimalusest, samuti võimalusest automaatselt skaleerida projekti jaoks eraldatud võimsust sõltuvalt serveri koormusest ja selle värskendamise lihtsusest sisseehitatud funktsioonide kaudu. *replit.com* pakub mugavat ja usaldusväärset keskkonda veebirakenduste arendamiseks ja juurutamiseks, mis sobib ideaalselt elektrimootori diagnostikasüsteemi jaoks [21].

1.2.7 Teadete saatmine

Teenus app.maileroo.com valiti e-posti teel teadete saatmiseks andurite normaalväärtusi ületavate näitude kohta. See teenus pakub võimalust luua meilide saatmiseks domeene ja kasutada seda minu lõputöö taustaserveris [22].

Maileroo pakub mugavaid tööriistu meiliteatiste seadistamiseks ja saatmiseks ning toetab ka domeene, mis võimaldab luua isikupärastatud saatja aadresse ja parandada meili edastamist.

See teenus on mugav ka selle poolest, et annab tasuta kasutamise võimaluse väikeprojektide puhul, lisaks on paindlik süsteem limiitide laiendamiseks, kui projekt laieneb ja päevas on vaja rohkem kirju.

1.2.8 Andmete kuvamine

Andurite andmete kuvamiseks valisin veebilehe javascripti kirjutamise, kasutades React.js [23] teeki.

React.js pakub mugavaid tööriistu interaktiivsete kasutajaliideste loomiseks. Selle peamised eelised andmete kuvamisel hõlmavad järgmist:

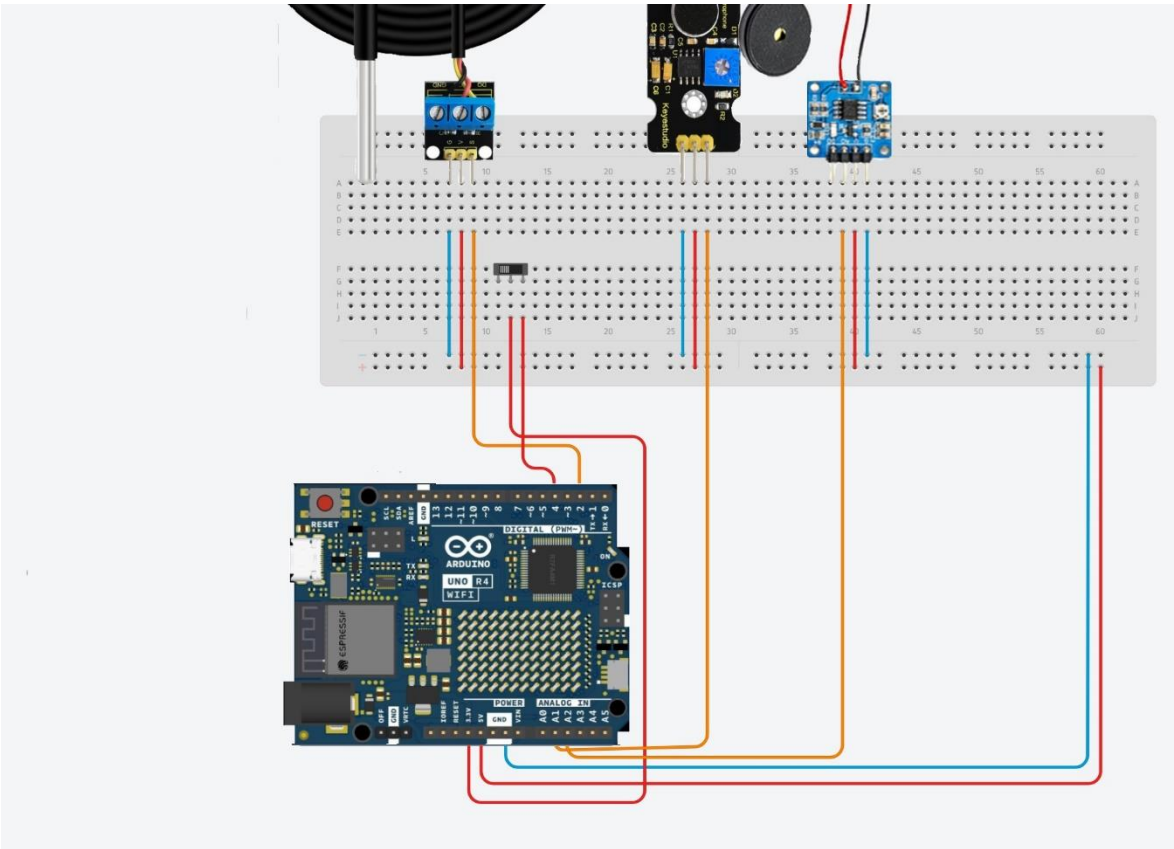
1. Komponendipõhine lähenemine: React.js võimaldab luua komponente, mis on korduvkasutatavad ja kergesti skaleeritavad. See on eriti kasulik suurte andmemahtude kuvamisel [23].
2. Virtuaalne DOM: React.js kasutab virtuaalset DOM-i, mis võimaldab tõhusalt värskendada ainult neid veebilehe osi, mis on muutunud. See parandab liidese jõudlust ja reageerimisvõimet andmete värskendamisel [23].
3. Lai kogukond ja tugi: React.js-il on tohutu kogukond arendajaid, kes on valmis vastama kõigile teie küsimustele. React.js-i abil arendamise hõlbustamiseks on palju teeke ja ressursse [23].

React.js'i valimine andmete kuvamiseks on tingitud selle paindlikkusest ja jõudlusest, mistõttu on see minu lõputöö jaoks suurepärane valik.

2 SÜSTEEMI ARENDUS JA TESTIMINE

2.1 Füüsilise prototüübi väljatöötamine ja kokkupanek

Prototüübi ehitamiseks kasutasin olemasolevaid ja odavaid komponente, et viia ellu lõputöö ideed odavast ja ligipääsetavast diagnostikast, mis põhineb Arduino mikrokontrolleril. Montaažiprotsess hõlmas temperatuuri-, vibratsiooni- ja heliandurite ühendamist Arduino-ga. Andurite paigaldamiseks kasutati leivatahvlit, millele asetati temperatuuri-, vibratsiooni- ja heliandurid. Andurid ühendati Arduino-ga järgmise skeemi järgi (Joonis 1). Temperatuurianduri signaali väljund Arduino sisendile number 2, helianduri signaal väljund Arduino analoogsisendile A1, vibratsioonianduri signaali väljund Arduino analoogsisendile A2. Andurite toiteallikaks oli 5V Arduino siin, mis oli kõigi anduritega ühendatud leivaplaadi kaudu. Võimalus panna Arduino "pausi" režiimi, et andmete lugemine ja andmebaasi sisestamine lõpetataks, toimub Arduino siini 3,3 V toiteploki lühistamiseks Arduino sisendiga number 4.



Joonis 1 Skeem andurite ühendamiseks Arduinoga [24], [25], [26], [27]

2.2 Tarkvara arendamine

2.2.1 Kontrollerile programmi kirjutamine

Arduino kontrolleri programmi arendamiseks kasutasin C++ programmeerimiskeelt ja Arduino IDE-d [28]. See kood esindab prototüübi põhiloogikat, sealhulgas anduritelt andmete lugemist ja nende edastamist serverisse edasiseks töötlemiseks.

Kõigepealt initsialiseeritakse enne koodi põhiosa vajalikud teegid, samuti kontrolleri füüsiliste väljundite ja sisendite toimimise installimine ja määramine (Joonis 1.2).

```
#include <OneWire.h>
#include <DallasTemperature.h>
#include <WiFiSSLClient.h>
#include <ArduinoJson.h>
#include <WiFi.h>

const char* ssid = "HUAWEI_E5576_F305";           // Wifi SSID
const char* password = [redacted];   // Wifi password
const char* host = "flask-lenale301.replit.app"; // Server host name
const int httpsPort = 443;                  // Port HTTPS
const char* url = "/send-sensor-data";      // URL

const int oneWireBus = 2; // Digital input pin for temperature sensor 2
const int vibrationSensorPin = A2; // Analog input pin for vibration sensor A2
const int soundSensorPin = A1; // Analog input pin for sound sensor A1
const int stopSendingPin = 4; // Digital input pin for "stop sending data pin" 4

OneWire oneWire(oneWireBus); //Use of libraries to convert temperature sensor readings to the degrees Celsius scale.
DallasTemperature sensors(&oneWire); //Use of libraries to convert temperature sensor readings to the degrees Celsius scale.
WiFiSSLClient client;  // SSL client for secure connections
```

Joonis 1.1 Nõutavate teekide lähtestamine

Setup() plokis lähtestatakse vajalikud komponendid, näiteks jadaport silumisinfo väljastamiseks ja ühendus Wi-Fi võrguga.

```
void setup() {
  Serial.begin(9600);
  pinMode(stopSendingPin, INPUT); // Set a pin to stop sending data as input

  connectToWiFi(); // Connecting to Wifi
  sensors.begin(); // Initializing the DS18B20 sensor
}
```

Joonis 1.2 Nõutavate komponentide lähtestamine

Seejärel toimub prototüübi põhisilmus funktsioonis loop(). See kontrollib, kas kontaktilt saadab signaali andmete saatmise peatamiseks. Signaali tuvastamisel andmete saatmine peatatakse, vastasel juhul loetakse andmed anduritelt ja saadetakse serverisse (Joonis 1.3).

```
void loop() {
  if (digitalRead(stopSendingPin) == HIGH) {
    Serial.println("Stop sending data. 5 volts detected on pin 4.");
    while (true) {
      // Do nothing and wait until 5 volts is removed from pin 4
      if (digitalRead(stopSendingPin) == LOW) {
        Serial.println("5 volts removed. Resuming sending data.");
        break;
      }
    }
  } else {
    if (WiFi.status() == WL_CONNECTED) { // Checking Wifi status
      // Read sensor data one by one
      sensors.requestTemperatures();
      sendSensorData("tempSensor", sensors.getTempCByIndex(0)); // Sending temperature sensor data
      delay(2000); // Delay for 2 seconds
      sendSensorData("vibrationSensor", analogRead(vibrationSensorPin)); // Sending vibration sensor data
      delay(2000); // Delay for 2 seconds
      sendSensorData("soundSensor", analogRead(soundSensorPin)); // Sending sound sensor data
    } else {
      Serial.println("Wi-Fi not connected. Reconnecting...");
      connectToWiFi(); // Reconnecting to Wifi if connection was lost
    }
  }

  // Delay before making another request
  delay(1000);
}
```

Joonis 1.3 loop() prototüübi põhisilmus

Funktsioon connectToWiFi() vastutab Arduino Interneti-ühenduse eest, kasutades sisseehitatud Wi-Fi moodulit (Joonis 1.4).

```
// Wifi connection function
void connectToWiFi() {
  Serial.println("Connecting to Wi-Fi...");
  WiFi.begin(ssid, password);

  // Waiting for connection
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  Serial.println("Connected to Wi-Fi");
}
```

Joonis 1.4 connectToWiFi() funktsioon

Funktsioon sendSensorData() vastutab andmete serverisse saatmise eest. See genereerib anduri andmetega JSON-objekti ja saadab selle HTTPS-protokolli kaudu (Joonis 1.5).

```
// Function for sending data to the server
void sendSensorData(const char* sensorName, float sensorValue) {
  // Create JSON document
  StaticJsonDocument<200> jsonDoc;
  jsonDoc["sensor_name"] = sensorName;
  jsonDoc["value"] = sensorValue;

  // Serialize JSON document to a string
  String jsonData;
  serializeJson(jsonDoc, jsonData);

  // Connect to the server
  if (client.connect(host, httpsPort)) {
    Serial.println("Connected to the server");

    // Send the HTTP POST request
    client.println("POST " + String(url) + " HTTP/1.1");
    client.println("Host: " + String(host));
    client.println("Content-Type: application/json");
    client.print("Content-Length: ");
    client.println(jsonData.length());
    client.println("Connection: close");
    client.println(); // End of headers
    client.println(jsonData);

    // Read all the lines of the reply from server and print them to serial monitor
    while (client.connected()) {
      String line = client.readStringUntil('\n');
      if (line == "\r") { // Check if the line is end of headers (empty line)
        Serial.println("Headers received, waiting for response...");
      } else {
        Serial.println(line);
      }
    }

    // Close the connection
    client.stop();
    Serial.println("Connection closed.");
  } else {
    Serial.println("Failed to connect to server");
  }
}
```

Joonis 1.5 funktsioon sendSensorData().

Seega on Arduino kontrolleri programm terviklik süsteem, mis suudab lugeda anduritelt andmeid ja edastada need edasiseks töötlemiseks ja analüüsiks kaugserverisse.

2.2.2 Serveripoolne arendamine

Kasutasin serveripoolse koodi kirjutamiseks *Pythoni* programmeerimiskeelt. Arenduskeskkonnaks valiti replit.com pakutav sisseehitatud koodiredaktor. Replit.com toimib ka serveri ja vastuvõetud andmete töötlemise hostina (Joonis 2.1).

Flask serveri koodi algus.

```
from flask import Flask, request, jsonify
import psycopg2
from datetime import datetime
from smtplib import SMTP
from email.mime.text import MIMEText
from email.mime.multipart import MIMEMultipart

app = Flask(__name__)
```

Joonis 2.1 Vajalike moodulite importimine

See koodilõik käivitab rakenduse Flask ja impordib vajalikud moodulid veebiserveri haldamiseks, PostgreSQL andmebaasi ja meili saatmiseks.

SMTP sätete konfigureerimine (Joonis 2.2).

```
# Maileroo SMTP configuration
SMTP_HOST = 'smtp.maileroo.com'
SMTP_PORT = 587
SMTP_USERNAME = '[redacted]maileroo.org'
SMTP_PASSWORD = '[redacted]'
RECEIVER_EMAIL = 'lenale301@gmail.com'
```

Joonis 2.2 SMTP sätted

See osa määrab SMTP parameetrid e-kirjade saatmiseks, nagu serveri aadress, port, kasutaja mandaadid ja saaja aadress.

Meili funktsioon (Joonis 2.3).

```python
# Function to send email
def send_email(sensor_name, value):
    try:
        # Setup the MIME Multipart message
        msg = MIMEMultipart()
        msg['From'] = SMTP_USERNAME
        msg['To'] = RECEIVER_EMAIL
        msg['Subject'] = 'Alert: Abnormal Data Detected'
        body = f'Abnormal data detected from sensor {sensor_name}: {value}'
        msg.attach(MIMEText(body, 'plain'))

        # Connect to the Maileroo SMTP server
        server = SMTP(SMTP_HOST, SMTP_PORT)
        server.starttls()
        server.login(SMTP_USERNAME, SMTP_PASSWORD)
        server.sendmail(SMTP_USERNAME, RECEIVER_EMAIL, msg.as_string())
        server.quit()

        print("Email sent successfully.")
    except Exception as e:
        print(f"Error sending email: {e}")
```

Joonis 2.3 Meili saatmise funktsioon

See funktsioon vastutab e-kirja saatmise eest, kui tuvastatakse ebatavalised anduri andmed.

Funktsioon andmete sisestamiseks andmebaasi (Joonis 2.4).

```python
# Function to insert data into database
def insert_data(schema, table, data):
    conn = None
    try:
        conn = psycopg2.connect(dbname="postgres",
                                user="postgres.[redacted],
                                password="[redacted],
                                host="aws-0-eu-central-1.pooler.supabase.com",
                                port="6543")
        cur = conn.cursor()
        cur.execute(data['query'], data['values'])
        conn.commit()
        cur.close()
        print(f"Data sent successfully to {schema}.{table}.")
    except Exception as e:
        print(f"An error occurred while attempting to insert data: {e}")
    finally:
        if conn:
            conn.close()
```

Joonis 2.4 Andmete sisestamise funktsioon

See funktsioon vastutab andmete sisestamise eest PostgreSQL-i andmebaasi.

Anduri normaalsete väärtuste määramine (Joonis 2.5).

```
# Define normal values for each sensor
normal_values = {'tempSensor': 80, 'vibrationSensor': 10, 'soundSensor': 130}
```

Joonis 2.5 anduri normaalsed väärtused

Siin on määratletud iga anduri normaalväärtused.

Andmete saatmise funktsioon anduritelt vastavasse andmebaasi tabelisse (Joonis 2.6).

```
# Function to send sensor data to the appropriate table
def send_sensor_data(sensor_name, value):
  current_time = datetime.now().strftime('%H:%M:%S')
  current_date = datetime.now().strftime('%Y-%m-%d')
  schema = 'public'
  table = f'{sensor_name}Data'
  data = {
      'query':
      f'INSERT INTO "{schema}"."{table}" (value, date, time) VALUES (%s, %s, %s);',
      'values': (value, current_date, current_time)
  }

  # Insert data to normal sensor data table
  insert_data(schema, table, data)

  # Check if value exceeds 20% of the normal value and insert to AbnormalData if it does
  if value > normal_values[sensor_name] * 1.2:
    abnormal_data = {
        'query':
        f'INSERT INTO "{schema}"."AbnormalData" ("sensorName", value, date, time) VALUES (%s, %s, %s, %s);',
        'values': (sensor_name, value, current_date, current_time)
    }
    insert_data(schema, 'AbnormalData', abnormal_data)
    send_email(sensor_name, value)
```

Joonis 2.6 Funktsioon andmete saatmiseks vastavasse tabelisse

See funktsioon saadab andmed anduritelt serverisse ja töötleb neid. Kui tuvastatakse ebanormaalsed väärtused, kirjutatakse need tabelisse AbnormalData ja saadetakse meil.

POST-i päringu töötleja (Joonis 2.7).

```python
@app.route('/send-sensor-data', methods=['POST'])
def receive_sensor_data():
  content = request.json
  if content is not None:
    sensor_name = content['sensor_name']
    value = content['value']

    # Use the function to send sensor data to the appropriate table
    send_sensor_data(sensor_name, value)

  # Return a response
  return jsonify({"message": "Data received successfully"}), 200
```

Joonis 2.7 POST-i päringute töötlemine

See kooditükk on serveri POST-päringute töötleja. See võtab anduritelt vastu andmeid, kutsub nende töötlemiseks funktsiooni send_sensor_data ja tagastab teate, mis näitab, et andmed on edukalt vastu võetud.

Rakenduse käivitamine (Joonis 2.8).

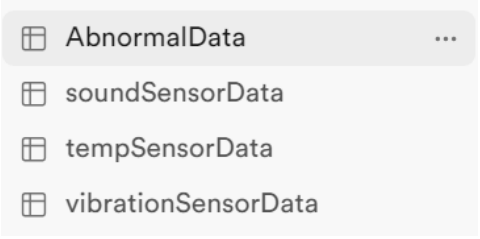
```python
if __name__ == '__main__':
  app.run()
```

Joonis 2.8 Rakenduse käivitamine

See kood käivitab rakenduse Flask, kui see käivitati otse ja seda ei imporditud moodulina.

2.2.3 Andmebaasi seadistamine

Andmete salvestamiseks kasutatakse lõputöös Supabase teenust, mis pakub mugavat ja skaleeritavat lahendust andmebaaside haldamiseks. Andmebaas koosneb neljast tabelist: AbnormalData, soundSensorData, tempSensorData ja vibrationSensorData (Joonis 3.1).



Joonis 3.1 Andmebaasi tabelid

SoundSensorData, tempSensorData ja vibrationSensorData tabelitel on järgmine struktuur:

id (andmetüüp: int8) - kirje kordumatu identifikaator,

väärtus (andmetüüp: float8) - vastavalt andurilt saadud väärtus,

kuupäev (andmetüüp: date) – andmete saamise kuupäev,

aeg (andmetüüp: time) - andmete vastuvõtmise aeg.

Tabelil AbnormalData on järgmine struktuur:

id (andmetüüp: int8) - kirje kordumatu identifikaator,

sensorName (andmetüüp: varchar) – anomaalse väärtuse salvestanud anduri nimi,

väärtus (andmetüüp: float8) - andurilt saadud väärtus,

kuupäev (andmetüüp: date) – anomaalia tuvastamise kuupäev,

aeg (andmetüüp: time) – anomaalia tuvastamise aeg.

See tabeli struktuur võimaldab tõhusalt korraldada andmete salvestamist, pakkudes neile mugavat juurdepääsu ja võimalust neid analüüsida. Järgnev on koodilõik tabelite loomiseks PostgreSQL-i andmebaasis (Joonis 3.2).

```
1   CREATE TABLE soundSensorData (
2       id SERIAL PRIMARY KEY,
3       value FLOAT8,
4       date DATE,
5       time TIME
6   );
7
8   CREATE TABLE tempSensorData (
9       id SERIAL PRIMARY KEY,
10      value FLOAT8,
11      date DATE,
12      time TIME
13  );
14
15  CREATE TABLE vibrationSensorData (
16      id SERIAL PRIMARY KEY,
17      value FLOAT8,
18      date DATE,
19      time TIME
20  );
21
22  CREATE TABLE AbnormalData (
23      id SERIAL PRIMARY KEY,
24      sensorName VARCHAR,
25      value FLOAT8,
26      date DATE,
27      time TIME
28  );
```

Joonis 3.2 Tabelite loomine andmebaasis

See kood loob vajalikud määratud struktuuriga tabelid PostgreSQL andmebaasis, tagades andmete korrektse salvestamise ja hilisema rakenduses kasutamise.

2.2.4 Veebiliidese loomine

Andurite andmete kuvamiseks kasutatakse React.js-i koos axiose teegiga, et töötada koos serveri API-ga.

Fail App.js määratleb rakenduse komponendi, mis on veebirakenduse põhikomponent.

Faili alguses imporditakse vajalikud sõltuvused, näiteks React, useState ja useEffect hook`id, samuti HTTP päringutega töötamiseks mõeldud aksiod ja App.css stiilifail (Joonis 4.1).

```
import React, { useEffect, useState } from 'react';
import axios from 'axios';
import './App.css';
```

Joonis 4.1 Nõutavate sõltuvuste importimine

Seejärel määratletakse rakenduse komponent. Selle komponendi sees kasutatakse komponendi oleku juhtimiseks useState olekuhook`e. sensorData salvestab andurite andmed ja showAll juhib andmete kuvamist tabelites (Joonis 4.2).

```
const App = () => {
  const [sensorData, setSensorData] = useState({});
  // Initialize the showAll state as an object where keys are table names and values are booleans
  const [showAll, setShowAll] = useState({});
```

Joonis 4.2 Rakenduse komponendi definitsioon

See koodiplokk kasutab hook`i useEffect, et täita komponendi paigaldamisel ja oleku showAll värskendamisel serverist andmete laadimise kõrvalmõju. FetchData sees teeb see serveri API-le asünkroonseid päringuid, kasutades aksiosid andmete toomiseks erinevatest lõpp-punktidest (Joonis 4.3).

```
useEffect(() => {
  const fetchData = async () => {
    const urls = [
      'soundSensorData',
      'tempSensorData',
      'vibrationSensorData',
      'AbnormalData'
    ];
    const headers = {
      'Content-Type': 'application/json',
      // Api key
      'apikey': '[redacted]
      'Authorization': '[redacted]
    };

    try {
      const requests = urls.map((url) =>
        axios.get(`https://oryjtlzlwaxrkruzkiqq.supabase.co/rest/v1/${url}`, { headers })
      );

      const responses = await Promise.all(requests);
      const data = responses.reduce((acc, response, index) => {
        acc[urls[index]] = response.data;
        return acc;
      }, {});

      setSensorData(data);
      // Initialize showAll state only if it is empty
      if (Object.keys(showAll).length === 0) {
        setShowAll(urls.reduce((acc, url) => {
          acc[url] = false; // Initially, do not show all items
          return acc;
        }, {}));
      }
    } catch (error) {
      console.error('Error fetching data:', error);
    }
  };

  fetchData();
  const interval = setInterval(fetchData, 5000);
  return () => clearInterval(interval);
}, [showAll]); // Add showAll as a dependency
```

Joonis 4.3 Päringute saatmine ja andmete allalaadimine serverist

Seda funktsiooni kasutatakse oleku showAll muutmiseks, kui kasutaja klõpsab tabelis täiendavate andmete kuvamiseks või peitmiseks nupul "Kuva rohkem" või "Kuva vähem" (Joonis 4.4).

```
const toggleShowAll = (tableName) => {
  setShowAll(prev => ({ ...prev, [tableName]: !prev[tableName] }));
};
```

Joonis 4.4 Tabeli muutmise funktsioon

Komponendi lõpus tagastatakse struktuur, mis kuvab andurite andmetega tabeleid. Igal tabelis on päis, põhiosa andmetega ja nupp lisaandmete kuvamiseks/peitmiseks (Joonis 4.5).

```
  return (
    <div className="App">
      {Object.keys(sensorData).length > 0 ? (
        Object.entries(sensorData).map(([tableName, tableData]) => (
          <div key={tableName} className="table-container">
            <h2>{tableName}</h2>
            <table className="table">
              <thead>
                <tr>
                  <th>ID</th>
                  <th>Value</th>
                  <th>Date</th>
                  <th>Time</th>
                </tr>
              </thead>
              <tbody>
                {(showAll[tableName] ? tableData : tableData.slice(-10)).map((item) => (
                  <tr key={item.id}>
                    <td>{item.id}</td>
                    <td>{item.value}</td>
                    <td>{item.date}</td>
                    <td>{item.time}</td>
                  </tr>
                ))}
              </tbody>
            </table>
            <button onClick={() => toggleShowAll(tableName)}>
              {showAll[tableName] ? 'Show Less' : 'Show More'}
            </button>
          </div>
        ))
      ) : (
        <p>Loading data...</p>
      )}
    </div>
  );
};

export default App;
```

Joonis 4.5 Andmetabelite kuvamise kood

Nüüd vaatame faili index.js, mis vastutab rakenduse juurkomponendi renderdamise eest (Joonis 4.6).

```
import React from 'react';
import ReactDOM from 'react-dom/client'; // Import createRoot
import './index.css';
import App from './App';

const root = ReactDOM.createRoot(document.getElementById('root')); // Using createRoot
root.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>
);
```

Joonis 4.6 Fail Index.js

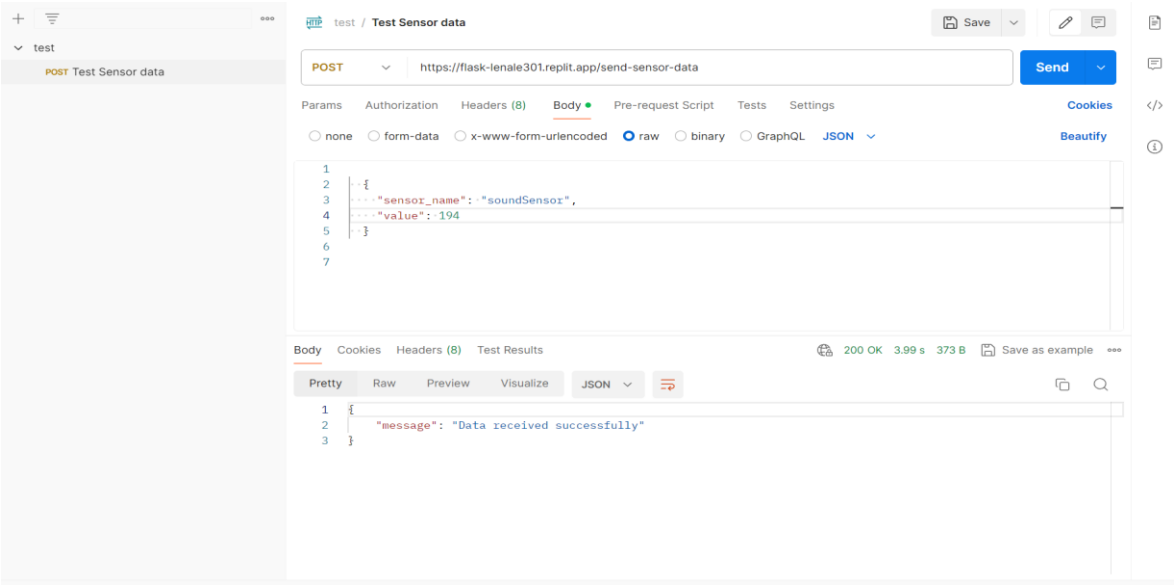
See kood loob juurrakenduse komponendi ja renderdab selle määratud DOM-i elemendile, kasutades ReactDOM-i createRoot meetodit. Rakenduse komponent reageerib olekumuutustele ja kuvab andurite andmed tabelites, võimaldades kasutajal juhtida andmete kuvamist nuppude "Näita rohkem" ja "Kuva vähem" abil.

2.3 Testimine ja silumine

Oma lõputöös arendamise ajal pöörasin erilist tähelepanu testimisele ja silumisele, et kõik süsteemi komponendid töötaksid õigesti.

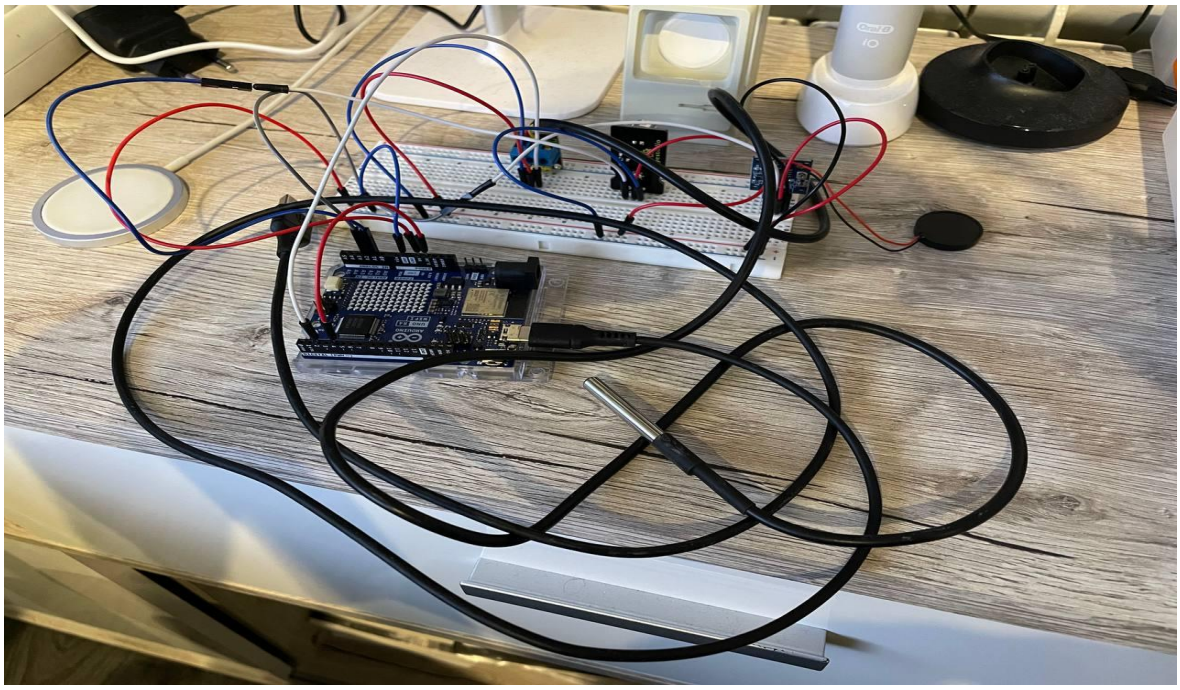
2.3.1 Funktsionaalsuse testimine

Rakenduse serveripoolse funktsionaalsuse testimiseks kasutasin Postmani programmi. Selle abiga saatsin serverisse erinevat tüüpi päringuid, et kontrollida, kas andmeid on õigesti töödeldud ja andmebaasiga suhelnud (Joonis 5.1).



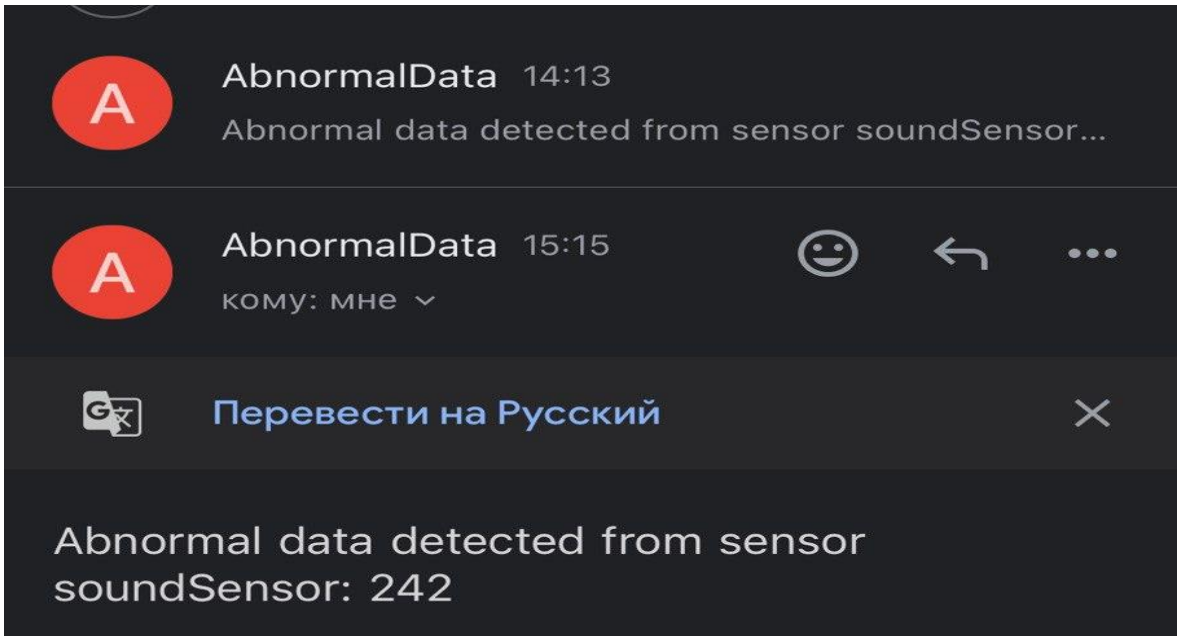
Joonis 5.1 Postman programm

Samuti kontrollisin kodus prototüüpi testides (Joonis 5.2), kuidas Arduino andmeid saadab ja kuidas server neid vastu võtab ja töötleb.



Joonis 5.2 Füüsiline prototüüp

Testisin ka veebilehte andmebaasist andmete vaatamiseks ja e-posti teenuse funktsionaalsust e-posti teadete saatmiseks anduri ebanormaalsete näitude korral (Joonis 5.3).



Joonis 5.3 Meiliteade

Peale esmaseid katsetusi kodus tegin ka lisatesti, paigaldades Arduino töötavale liftile. Andurid paigaldati lifti vintsi mootorile ja nende näidud edastati süsteemi. See võimaldas testida süsteemi reaalsetes tingimustes ning tuvastada võimalikud keskkonna ja seadmete tegeliku tööga seotud probleemid (Joonis 5.4).

Joonis 5.4 Prototüüp lifti vintsi mootoril

Arduino ühendamiseks Internetiga selles testimises kasutati LTE-modemi, mis võimaldab luua Interneti-ühenduse peaaegu igas olukorras (Joonis 5.5).

Joonis 5.5 LTE-modem

Funktsionaalsuse testimise tulemusena oli võimalik kontrollida selle funktsionaalsust, samuti parandada tuvastatud puudusi ja probleeme.

2.3.2 Silumine ja tõrkeotsing

Süsteemi jõudluse testimise käigus tuvastati mõned probleemid, mis nõudsid silumist ja koodi muutmist.

Arduino Interneti-ühenduse probleem

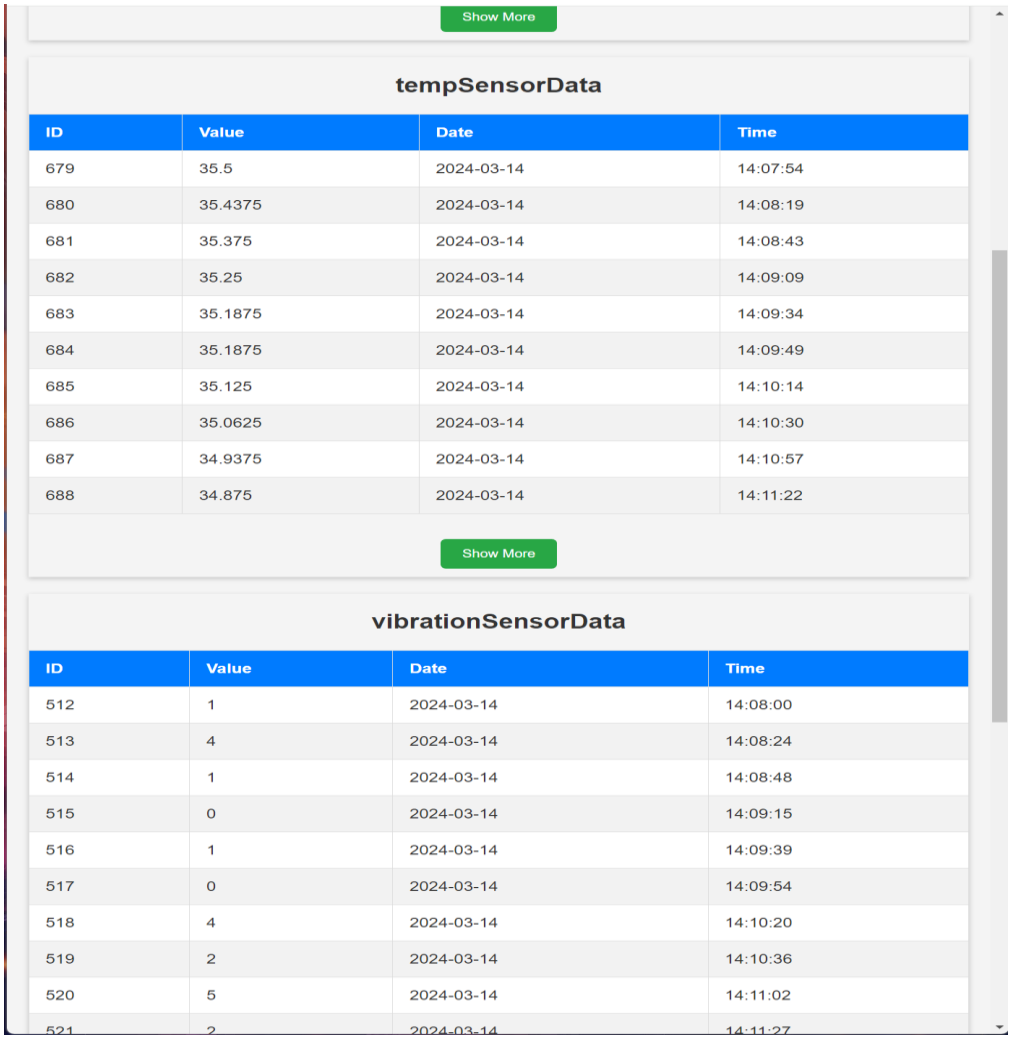
Üks tuvastatud probleeme oli Arduino Interneti-ühenduse ebastabiilsus Wi-Fi kaudu. Analüüsi tulemusena leiti, et Arduino võib pärast mõnda aega töötamist Wi-Fi võrgust spontaanselt lahti ühendada. Selle probleemi lahendamiseks tehti muudatusi Arduino koodis.

Arduino koodi algses versioonis ühendus seade Wi-Fi võrku ainult sisselülitatuna ja kui Wi-Fi signaal töö käigus kadus, ei saanud see võrguga uuesti ühendust. Programmi uues versioonis kontrollib Arduino Wi-Fi-ühendust iga kord, kui saadab anduritelt andmepaketi. Kui tuvastatakse ühenduse puudumine, kutsub Arduino funktsiooni Wi-Fi-võrguga uuesti ühenduse loomiseks. Pärast Interneti-ühenduse taastamist jätkab Arduino andmete saatmist.

Probleem andmete kuvamisel veebilehel

Veebiliidese testimisel avastati probleem seoses vajadusega pikka aega lehte kerida, kui andmebaasi tabelites oli palju andmeid. Kasutajakogemuse parandamiseks on tehtud muudatusi veebilehe koodis.

Kõik tabelid kuvatakse nüüd vaikimisi ahendatud kujul, näidates ainult andmebaasi viimaseid ridu. Vajadusel saab kasutaja tabelit täielikult laiendada, kasutades tabeli allosas asuvat spetsiaalset nuppu. Samuti lisati "intervall" meetod, mis võimaldab automaatselt uuendada teavet tabelites teatud ajavahemike järel, mis väldib vajadust lehekülge käsitsi uuesti laadida, et kuvada anduritelt uusi andmeid (Joonis 5.6).



Joonis 5.6 Veebileht andmetega tabelite kuvamiseks

2.4 Saadud andmete analüüs

Pärast seiresüsteemi prototüübi paigaldamist töötava lifti vintsmootorile salvestati andmebaasi lifti töötamise ajal näidud. Nende andmete analüüs võimaldas määrata andurite signaalide normaalseid ja ebanormaalseid väärtusi.

Heliandur

Maksimaalne helianduri tuvastatud väärtus oli 242. See väärtus saadi siis, kui lift liikus vähendatud kiirusega ja vintsi mootor ümises valkjult. Kontrollimisel leiti, et kunstlikult tekitatud müra, näiteks vali muusika või karjumise korral annab andur väärtusi kuni 400. Anduri keskmised näidud ei ületanud 50, kuid mootori käivitamisel võisid need ulatuda 100-ni. -120. Tuleb märkida, et väärtusi, mis ei ületa 130, peetakse normaalseks. Selle väärtuse ületamine 20% võib viidata võimalikule ebanormaalsele toimingule ja need näidud saadetakse tabelisse AbnormalData ja meilisõnum. (Joonis 6.2).

id int8	value float8	date date	time time
390	242	2024-03-14	13:15:32
391	0	2024-03-14	13:16:22
392	0	2024-03-14	13:16:51
393	4	2024-03-14	13:17:09
394	2	2024-03-14	13:17:47
395	30	2024-03-14	13:18:03
396	0	2024-03-14	13:18:18
397	0	2024-03-14	13:18:32
398	0	2024-03-14	13:19:53
399	7	2024-03-14	13:20:10
400	9	2024-03-14	13:21:19
401	3	2024-03-14	13:21:46
402	23	2024-03-14	13:22:35
403	135	2024-03-14	13:22:51

Joonis 6.1 Andmed tabelis soundSensorData

id int8	sensorName varchar	value float8	date date	time time
68	soundSensor	242	2024-03-14	13:15:32

Joonis 6.2 Näidisandmed tabelis AbnormalData

Vibratsiooniandur

Maksimaalne vibratsiooniandurи tuvastatud väärtus oli 12. See juhtus hetkel, kui mootor hakkas vibreerima liftikabiini väikesele kiirusele ülemineku tõttu. Katse käigus selgus, et kunstlikult tekitatud tugeva vibratsiooni korral võis vibratsiooniandur anda väärtusi kuni 400. Keskmised väärtused ei ületanud 10, enamus oli 5 ringis. vibratsiooniandurи väärtus oli 10 või vähem. Selle väärtuse ületamine 20% võib viidata võimalikule ebanormaalsele toimingule ja need näidud saadetakse tabelisse AbnormalData ja meilisõnum. (Joonis 6.3).

id int8	value float8	date date	time time
273	3	2024-03-14	12:12:09
274	2	2024-03-14	12:12:26
275	2	2024-03-14	12:12:42
276	2	2024-03-14	12:13:18
277	10	2024-03-14	12:13:36
278	3	2024-03-14	12:13:53
279	5	2024-03-14	12:14:30
280	6	2024-03-14	12:14:44
281	6	2024-03-14	12:14:59
282	4	2024-03-14	12:15:14
283	12	2024-03-14	12:15:39
284	4	2024-03-14	12:15:54
285	0	2024-03-14	12:16:10
286	1	2024-03-14	12:16:26
287	4	2024-03-14	12:16:42

Joonis 6.3 Andmed vibrationSensorData tabelis

Temperatuuriandur

Maksimaalne temperatuurianduri näit oli 51,9375 kraadi. Arvestades talvel madalat temperatuuri lifti masinaruumis, võeti normaalväärtuseks 80 kraadi ja alla selle. Selle väärtuse ületamine 20% võib viidata võimalikule mootori ülekuumenemise ohule ja see kajastub tabelis AbnormalData (Joonis 6.4).

id int8	value float8	date date	time time
363	50.375	2024-03-14	12:19:02
364	51.0625	2024-03-14	12:19:25
365	51.5	2024-03-14	12:19:47
366	51.625	2024-03-14	12:20:07
367	51.6875	2024-03-14	12:20:23
368	51.8125	2024-03-14	12:20:38
369	51.9375	2024-03-14	12:20:58
370	51.9375	2024-03-14	12:21:20
371	51.875	2024-03-14	12:21:34
372	51.875	2024-03-14	12:21:50
373	51.75	2024-03-14	12:22:07
374	51.75	2024-03-14	12:22:24
375	51.6875	2024-03-14	12:22:39
376	51.5	2024-03-14	12:23:00

Joonis 6.4 Andmed tempSensorData tabelis

KOKKUVÕTE

See lõputöö tõstab esile liftivintsmootorite seisukorra jälgimise süsteemi väljatöötamist, mis annab olulise panuse liftisüsteemide ohutuse ja töökindluse parandamisse. Töö hõlmab olemasolevate diagnostikasüsteemide analüüsi, seiresüsteemi komponentide valikut ja seadistamist, kontrolleri ja serveri tarkvara väljatöötamist ning veebiliidese loomist andmete kuvamiseks. Süsteemi testimine reaalsetel seadmetel kinnitas selle tõhusust ja näitas võimalusi edasiseks täiustamiseks ja kohandamiseks liftide konkreetsete töötingimustega.

Saadud andmete põhjal saab teha järgmised järeldused:

1. Süsteemi teostus ja efektiivsus: Arduino ja erinevate andurite (temperatuur, vibratsioon, heli) kasutamisel põhinev lifti vintsi mootorite seisukorra jälgimise süsteem demonstreerib kõrget efektiivsust mootorite võimalike rikete ja rikete ennetamisel. Seda kinnitavad läbiviidud testid ja kogutud andmete analüüs.

2. Tehnoloogilised aspektid: Arduino platvormi kasutamine ja sobivate andurite valik tagas süsteemi paindlikkuse ja mastaapsuse, võimaldades seda kohandada erinevatele liftide mudelitele ja töötingimustele. Väljatöötatud tarkvara töötleb tõhusalt andurite andmeid, tagades nende täpsuse ja asjakohasuse.

3. Praktiline tähtsus: Seiresüsteemi kasutuselevõtt võib oluliselt tõsta liftisüsteemide ohutust ja töökindlust, lühendada diagnostikale ja remondile kuluvat aega ning samuti ennetada mootoririketega seotud hädaolukordi.

4. Arenguväljavaated: Edasised uuringud võivad olla suunatud andmetöötlusalgoritmide täiustamisele, arenenumate diagnostikameetodite väljatöötamisele, samuti süsteemi integreerimisele olemasolevate hoonete haldus- ja monitooringsüsteemidega, et luua ühtne platvorm liftide ohutuse ja ekspluatatsiooni haldamiseks.

Üldiselt on väljatöötatud süsteem oluline samm liftisüsteemide tõhususe ja ohutuse parandamise suunas. See demonstreerib kaasaegsete tehnoloogiate võimekust monitooringu ja diagnostika vallas, pakkudes lahendusi aktuaalsetele liftiehituse ja liftide käitamise probleemidele.

SUMMARY

Thesis Title: "Fault monitoring of elevator winch motors", author: Aleksandr Lendi

Introduction:
This thesis explores the design and development of a diagnostics and monitoring system for elevator motors, showing the importance of timely maintenance and effective fault detection to enhance operational safety and efficiency. The creation of a reliable and affordable diagnostic system based on Arduino plays an important role in increasing the reliability of elevator motors.

Objective:
The objective of this study is to devise a robust monitoring system that can help to identifying early signs of elevator motor failures, thereby minimizing downtime and maintenance costs.

Importance and Relevance:
With elevators being important in modern infrastructure, ensuring their uninterrupted operation is vital. An efficient and reliable diagnostic system not only safeguards against potential failures but also extends the operational life of elevator components.

Tasks of the Thesis:

1. Analysis of existing diagnostic technologies for elevator motors.
2. Development and testing of an Arduino-based prototype to monitor key winch motors parameters.
3. Development of server and user parts of the monitoring system.
4. Evaluation of system effectiveness through practical experiments and data analysis.

Completed work:

1. Component Selection and System Design: A detailed comparative analysis of existing diagnostic technologies was performed, leading to the selection of Arduino as the primary platform due to its affordability and versatility. Key sensors were chosen, including temperature, vibration, and sound sensors, to monitor parameters of elevator winch motors.
2. Development of the Prototype: An Arduino-based prototype was developed, integrating the selected sensors. The physical assembly involved connecting the

sensors to the Arduino board. A detailed schematic was created to ensure proper connectivity and functionality.

3. Software Development: Custom software was developed for both the Arduino controller and the server side. The Arduino program was written in C++ to read sensor data and send it to a remote server. On the server side, a Flask-based application was developed using Python to receive data, store it in a PostgreSQL database, and provide a web interface for real-time monitoring.
4. Testing and Validation: The system underwent testing, first in a controlled environment and then on an actual elevator winch motor. The tests focused on the system's ability to read and transmit sensor data, detect anomalies, and generate alerts. Adjustments were made based on the test results to enhance reliability and performance.
5. Data Analysis and Evaluation: Collected data was analyzed to determine normal operating ranges. The effectiveness of the system was evaluated based on its ability to detect deviations from normal operation and its overall impact on maintenance strategies.

Conclusion:

This thesis successfully demonstrates the potential of Arduino-based systems of monitoring in improving the diagnostics and monitoring processes of elevator motors. The findings suggest that such technologies can reduce the incidence of faults and enhance maintenance strategies, ensuring safer and more reliable elevator operations.

KASUTATUD KIRJANDUSE LOETELU

1) Flavio Solon Araujo Dutra, Rafael Souza de Oliveira, Thiago Matos Brazil, Jean Mark Lobo de Oliviera, Pablo Augusto da Paz Elleres. Real-time monitoring of industrial induction motors by current and vibration analysis, (November 2023) [Online]
https://www.researchgate.net/publication/377022269_REAL-TIME_MONITORING_OF_INDUSTRIAL_INDUCTION_MOTORS_BY_CURRENT_AND_VIBRATION_ANALYSIS
(20.04.2024)

2) Vladimir Veselkov, Maxym Zhalo, Andrei Turygin, Milos Hammer, Robert Jankovych. Contribution to diagnostics of asynchronous motors, (December 2015) [Online]
https://www.researchgate.net/publication/287157071_Contribution_to_diagnostics_of_asynchronous_motors
(20.04.2024)

3) Catra indra Cahyadi, Panangian Mahadi Sihombing, Hairul Amren Samosir, Usman. An IoT prototype for temperature monitoring and automatic control of electric motor, (September 2023) [Online]
https://www.researchgate.net/publication/373976073_AN_IOT_PROTOTYPE_FOR_TEMPERATURE_MONITORING_AND_AUTOMATIC_CONTROL_OF_ELECTRIC_MOTOR
(21.04.2024)

4) Arduino Uno R4 Wi-Fi [Online]
https://store.arduino.cc/products/uno-r4-wifi
(05.02.2024)

5) Omron K6CM-CI2M Motor Condition Monitoring Device [Online]
https://industrial.omron.eu/en/products/k6cm-ci2m
(05.02.2024)

6) PowerLogic™ ION9000 meter [Online]
https://www.se.com/us/en/product/METSEION92040/powerlogic-ion9000-meter-din-mount-192-mm-display-b2b-adapter-hw-kit/?range=64241-powerlogic-ion9000-series&selectedNodeId=12367169875
(05.02.2024)

7) ABB Ability™ Digital Powertrain - Condition Monitoring for rotating equipment [Online]
https://new.abb.com/motors-generators/motoren-generatorenservice/advanced-services/smart-sensor
(05.02.2024)

8) Condition monitoring of rotating equipment fitted with ABB AbilityTM Smart Sensors [Online]
https://search.abb.com/library/Download.aspx?DocumentID=9AKK107799&Language

Code=en&DocumentPartId=&Action=Launch
(05.02.2024)
9) Arduino Uno Rev3 [Online]
https://store.arduino.cc/products/arduino-uno-rev3
(11.02.2024)
10) Raspberry Pi Pico and Pico W [Online]
https://www.raspberrypi.com/documentation/microcontrollers/raspberry-pi-pico.html
(11.02.2024)
11) Raspberry Pi Pico WH [Online]
https://raspberrypi.dk/en/product/raspberry-pi-pico-wh/?src=raspberrypi
(11.02.2024)
12) Keyestudio Analog Sound Noise Sensor Detection Module for Arduino [Online]
https://www.keyestudio.com/products/free-shipping-keyestudio-analog-sound-noise-sensor-detection-module-for-arduino
(14.02.2024)
13) Gravity: Analog Sound Sensor For Arduino [Online]
https://www.dfrobot.com/product-83.html
(14.02.2024)
14) Electret Microphone Amplifier - MAX4466 with Adjustable Gain [Online]
https://www.adafruit.com/product/1063
(14.02.2024)
15) DHT22 Datasheet(PDF) [Online]
https://html.alldatasheet.com/html-pdf/1132459/ETC2/DHT22/225/2/DHT22.html
(14.02.2024)
16) Keyestudio DS18B20 Waterproof /Stainless steel Temperature Detector Sensor With Module [Online]
https://www.keyestudio.com/products/keyestudio-ds18b20-waterproof-stainless-steel-temperature-detector-sensor-with-module-black-and-eco-friendly
(14.02.2024)
17) Vibration Switch Sensor Module- Vibration Tapping Sensor [Online]
https://www.circuits-diy.com/vibration-switch-sensor-module-vibration-tapping-sensor/
(14.02.2024)
18) KY-002 VIBRATION SWITCH MODULE [Online]
https://arduinomodules.info/ky-002-vibration-switch-module/
(14.02.2024)
19) Database host service [Online]
https://supabase.com/

(02.03.2024)
20) Flask’s documentation [Online]
https://flask.palletsprojects.com/en/3.0.x/
(02.03.2024)
21) Flask server host service [Online]
https://replit.com/
(06.03.2024)
22) Mail sending domains service [Online]
https://app.maileroo.com/
(06.03.2024)
23) The library for web and native user interfaces [Online]
https://react.dev/
(15.03.2024)
24) Arduino Uno R4 Wi-fi.png [Online]
https://forum.fritzing.org/uploads/default/original/3X/6/d/6d3d54c019d239a22da986a9007f27be4d9e6288.png
(23.03.2024)
25) Keyestudio sound sensor.png [Online]
https://wiki.keyestudio.com/images/3/36/Ks0035%281%29.png
(23.03.2024)
26) Keyestudio temperature sensor.png [Online]
https://eckstein-shop.de/media/image/product/20170/lg/ks0316.webp
(23.03.2024)
27) Piezoelectric vibration sensor.png [Online]
https://electropeak.com/learn/wp-content/uploads/2021/02/Piezoelectric-Vibration-Required-Materials.jpg
(23.03.2024)
28) Arduino IDE guide [Online]
https://docs.arduino.cc/software/ide/
(02.03.2024)