TALLINNA TEHNIKAÜLIKOOL
INSENERITEADUSKOND

Elektroenergeetika ja mehhatroonika instituut

MÕÕTELAHENDUS JA SEADE AUTOMAATKÄIGUKASTI JUHTSIGNAALIDE SALVESTAMISEKS JA TAASESITAMISEKS

SIGNAL LOGGING AND PLAYBACK DEVICE FOR AUTOMATIC GEARBOXES

BAKALAUREUSETÖÖ

Üliõpilane: Timothy Rähmonen

Üliõpilaskood: 155493MAHB

Juhendaja: Raivo Sell

Tallinn, 2019

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.

“21 ” mail 2019

Autor: .....

/ allkiri /

Töö vastab bakalaureusetöö/magistritööle esitatud nõuetele

“.....”.....201… .

Juhendaja: ……

/ allkiri /

Kaitsmisele lubatud

“.....”.....201… .

Kaitsmiskomisjoni esimees .....

/ nimi ja allkiri /

LÕPUTÖÖ ÜLESANNE

Üliõpilane: Timothy Rähmonen, 155493MAHB.....(nimi, üliõpilaskood)

Õppekava, peaeriala: MAHB, Mehhatrooonika.....(kood ja nimetus)

Juhendaja(d): vanemteadur, Raivo Sell, 620 3268.....(amet, nimi, telefon)

Lõputöö teema:

(eesti keeles) *Mõõtelahendus ja seade automaatkäigukasti juhtsignaalide salvestamiseks ja taasesitamiseks*

(inglise keeles) Signal logging and playback device for automatic gearboxes

Lõputöö põhieesmärgid:

1. Mõõteseade peab võimaldama 3 PWM signaali, temperatuuri, rõhu ja kiiruse logimist SD kaardile vähemalt 1 minut
2. Peab olema võimalus lisada kommentaare salvestamise ajal ning salvestatud fail peab ühilduma Valve Body teststendi andmefailiga
3. Kuvada kõike signaale ekraanil reaalajas
4. Salvestatud fail peab olema taas mängitav teststendi

Lõputöö etapid ja ajakava:

Nr	Ülesande kirjeldus	Tähtaeg
1.	ATM käigukasti signaalide analüüs ja mõõtelahenduse kontseptsiooni loomine	05.03.2019
2.	Prototüüp seadme koostamine ja programmeerimine	15.03.2019
3.	Seadme testimine töösituatsioonis	20.03.2019
4.	Tarkvara arendus ja lahenduse viimistlus	15.04.2019

Töö keel: Eesti **Lõputöö esitamise tähtaeg:** “21” mai 2019a

Üliõpilane: Timothy Rähmonen /Digiallkirjastatud/ “28” veebruar 2019a

Juhendaja: Raivo Sell /Digiallkirjastatud/ “28” veebruar 2019a

SISUKORD

LÕPUTÖÖ ÜLESANNE 3

EESSÕNA 6

LÜHENDITE JA TÄHISTE LOETELU 7

SISSEJUHATUS 8

1. ÜLEVAADE SÜSTEEMIST 9

1.1 Automaatkäigukast 9

1.2 Täisautomaatsed käigukastid 9

1.2.1 Robotkäigukast 10

1.2.2 CVT-automaatkäigukast 11

1.3 Planetaar-automaatkäigukast 11

1.3.1 Hüdrodünaamiline pöördemomendi muundur 12

1.3.2 Planetaarmehhanism 13

1.3.3 Hüdraulika plokk 14

2. EELTEGEVUSED 15

2.1 Ülesande püstitus 15

2.2 Signaalide analüüs 15

2.2.1 Solenoidi PWM signaal 15

2.2.2 Käigukasti sisend- ja väljundkiirus 17

2.2.3 Käigukastiõli temperatuur 18

3. ELEKTROONIKA 20

3.1 Juhtplaat koos signaalide muundamise plaadiga 20

3.1.1 Juhtplaat 20

3.1.2 Signaalide muundamise plaat 21

3.2 Signaalide võimendusplaat 24

4. TARKVARA 28

4.1 Programmeerimine 28
4.1.1 Ülevaade FATFS moodulist 28
4.1.2 Programmi algoritm 29
4.2 Genereeritud faili ühildamine Valve Body teststendiga 32
5. MEHAANIKA 34
5.1 Juhtplaadi ja signaali muunduri korpus 34
5.2 Võimendusplaadi korpus 35
6. KATSED 37
6.1 Katsed laboritingimustes 37
6.2 Katsed autol 37
6.2.1 Katse tulemused 38
KOKKUVÕTE 39
SUMMARY 40
7. Kasutatud kirjandus 41
LISAD 44
LISA 1 Pildid juhtplaadist ning kasutajaliidesest 45
LISA 2 Pilt võimendusplaadist 46
LISA 3 Katkestuste kood juhtplaadil 47
LISA 4 PWM signaali lugemise ja genereerimise kood 49
LISA 5 Signaali lugemise alamfunktsioon 51
LISA 6 Signaali genereerimise alamfunktsioon 52

EESSÕNA

Käesolev lõputöö teema on saadud juhendaja vanemteadur Raivol Selli poolt. Töö teostamine toimub ettevõtte Hõbenool OÜ soovil.

Autor tänab kõiki, kes aitasid kaasa lõputöö valmimisele.

LÜHENDITE JA TÄHISTE LOETELU

ADC - analoog-digitaalmuundur (*Analog-to-Digital Converter*)

CVT - astmeteta käigukast ehk variaatorkäigukast (*Continuously Variable Transmission*)

CSV – komadega eraldatud väärtus (*Comma Seperated Values*)

GPIO – sisend-väljundviigud (*General-Purpose Input/Output)*

IDE - integreeritud programmeerimiskeskkond (*Integrated development environment*)

MOSFET – isoleeritud paisuga väljatransistor (*Metal-Oxide-Semiconductor Field-EffectTransistor*)

XML - laiendatav märgistuskeel (*Extensible Markup Language*)

SISSEJUHATUS

Tänapäeval kasutatakse automaatkäigukasti hüdraulika ploki vea tuvastamiseks teststende [1], see aga nõuab autolt automaatkäigukasti eemaldamist ning see protsess on aeganõudev. Teststend imiteerib auto juhtseadet ning võib tuvastada vea, kuid paraku ei anna see alati korrektset tulemust. Stend võib anda samad tulemused nii tervele kui ka vigasele käigukastile. Sama tulemus võib esineda juhul kui, auto juhtseade ja teststendi auto juhtseadet imiteeriv loogika ei ole kattu. Teststendi ja auto juhtseade ühildamiseks on vaja salvestada kõikvõimalikud auto juhtseadet mõjutavad signaalid.

Lõputöö eesmärk ongi salvestada konkreetse auto signaalid ning need andmed ühildada pärast katsestendis juhtseadet imiteeriva mõõteploki andmetega. See võimaldab esmase diagnostika teha ilma, et tuleks reaalselt käigukast eemaldada.

Antud töö esimeses pooles antakse ülevaade automaatkäigukastist. Põhjalikumalt on lahti seletatud uuritava käigukasti tüübi tööpõhimõte, et mõista, mis signaalide lugemise ja salvestamisega töös tegeletakse. Lühivaade on ka antud teistest automaatkäigukasti tüüpidest, et mõista erinevate käigukastide tööpõhimõtteid. Järgnevalt tutvustatakse lühidalt erinevaid signaalide tüüpe ning teostatakse nende signaalide analüüs. Selle analüüsi baasil hakatakse välja arendama mõõtelahendust.

Töö teine osa keskendub seadme valmimisele. Seletatakse lahti üldine elektroonika skeem, kuidas erinevad seadmed omavahel signaale jagavad. Tutvustatakse komponente, mis on kasutusel antud mõõtelahenduse tegemisel. Nende komponente kasutatakse elektriskeemides, mis on koostatud Altium Designer tarkvaraga. Pärast elektroonika tutvustamist antakse ülevaade tarkvaralisest lahendusest ning seletatakse lahti juhtplaadi üldine algoritm. Tarkvara koostamisel kasutati STM32CubeMX ja Atollic TrueStudio programme. Mõlemale plaadile modelleeriti korpused kasutades Solidworks programmi.

1. ÜLEVAADE SÜSTEEMIST

1.1 Automaatkäigukast

Käigukasti eesmärk on mootori pöördemomendi ja pöörlemiskiiruse ülekandmine veoratastele ning nende suuruste suurendamine või vähendamine. Automaatkäigukasti eripära on see, et käiku vahetatakse ülekandesuhte muutmiseks automaatselt. Automaatkäigukastid jaotatakse kaheks:

- Poolautomaatne käigukast
- Täisautomaatne käigukast

Poolautomaatne ja täisautomaatne käigukast erinevad käigu vahetamise meetodist. Poolautomaatse käigukasti puhul vahetatakse käiku, ehk muudetakse ülekandesuhet ja pöörlemiskiirust, käsitsi, kuid käiguvahetus käigukastis toimub automaatse siduriga. Täisautomaatne käigukast vahetab käike automaatselt elektrohüdraulilise või elektropneumaatilise juhtsüsteemi abil [2]. Antud lõputöö keskendub täisautomaat käigukastile.

1.2 Täisautomaatsed käigukastid

Täisautomaatseid käigukaste jaotatakse kolmeks põhiliseks kasutusel, mis on toodud välja **Tõrge! Ei leia viiteallikat.** koos nende erinevustega [2].

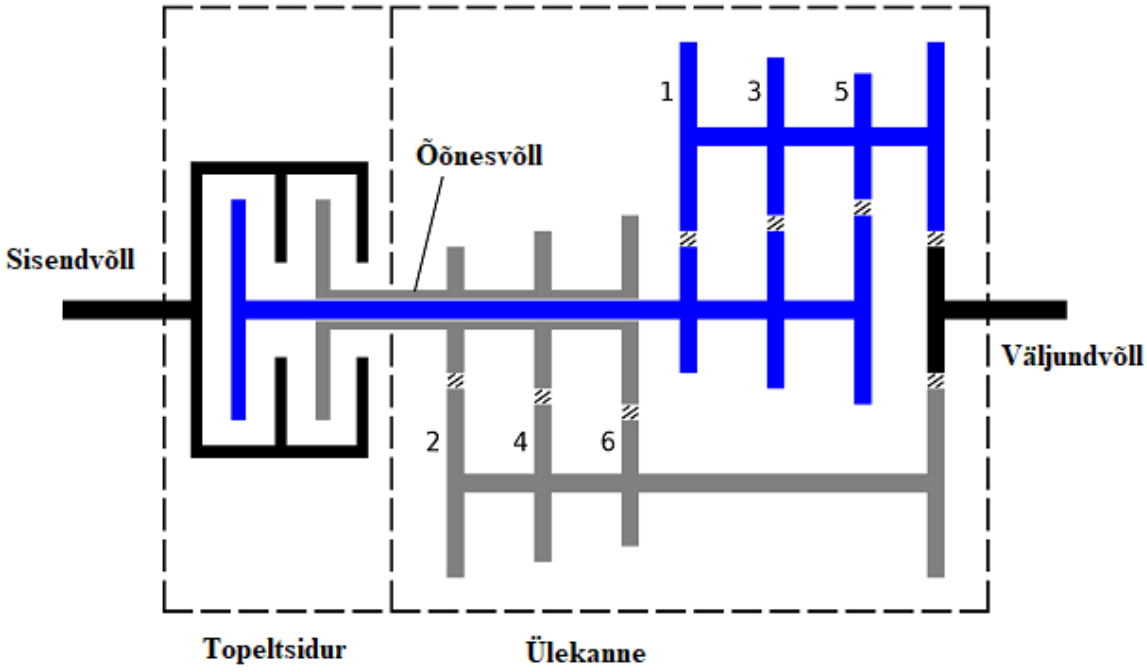
Tabel 1.1 Täisautomaat käigukastide erinevad tüübid

Täisautomaatne käigukast	Robotkäigukast	Planetaar-automaatkäigukast	CVT-automaatkäigukast
Jõuülekande	Membraanvedrusidur	Hüdrodünaamilise pöördemomendi muundur	Hüdrodünaamilise pöördemomendi muundur või membraanvedrusidur
Ülekande tüüp	Silinderhammasratas-ülekanne	Planetaarülekanne	Terasest tõukelülilint või plaatveokett
Ülekande suhte muutmine	Astmeline	Astmeline	Sujuv

Lõputöös uuritavaks automaatkäigukastiks on planetaar-automaatkäigukast. Sellest tingituna seletatakse planetaar-automaatkäigukasti tööpõhimõte üksikasjalikumalt lahti ning antakse lühi ülevaade robotkäigukastist ning CVT-automaatkäigukastist.

1.2.1 Robotkäigukast

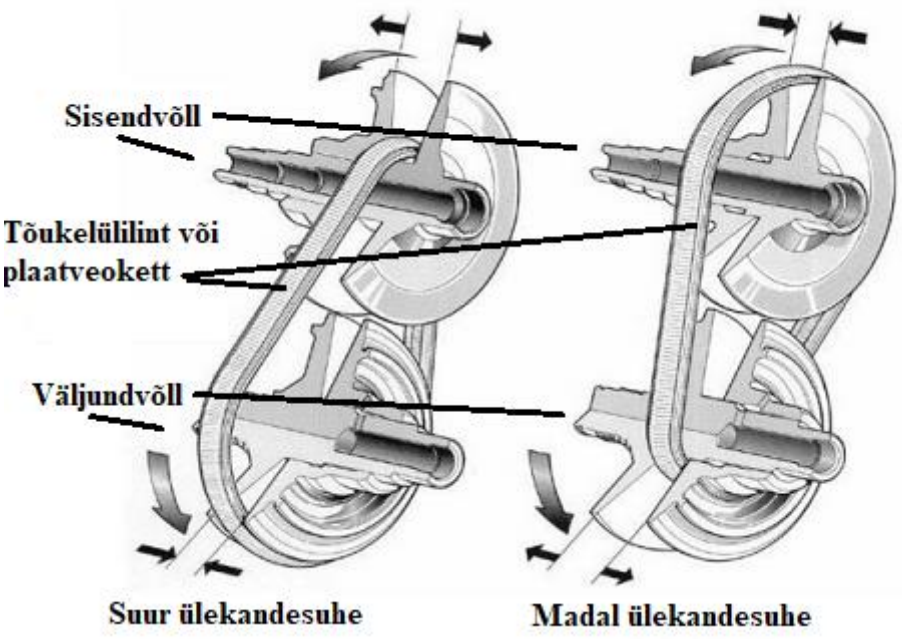
Robotkäigukasti all mõeldakse topeltsiduriga käigukasti. Robotkäigukast on täisautomaatne käigukast, mille tavapärast viie- või kuuekäigulist käigukasti lülitatakse siduriga automaatselt. Robotkäigukastis toimub siduri lülitamine ja käiguvahetus elektrohüdraulilselt. Käigukast toimib kui topelt manuaalne käigukast, kuid kõik käikude vahetused tehakse automaatselt toetudes sõidukiiruse, valikuhoova asendi, valitud sõidurežiimi ja gaasipedaali asendile. Optimaalseks lahutamiseks ja ühendamiseks kasutatakse mootori pöörlemiskiirust, käigukasti sisendvõlli pöörlemiskiirust ja siduriteekonna mõõtmiseks andureid. Käigu lülitamiseks kasutatakse valikusilindrit ja lülitussilindrit. Kasutatakse kahte sidurit, mis on ühendatud kahe erineva sisendvõlli külge. Esimesel sisendvõllil asuvad paaritud käigud 1, 3, 5 ja tagurpidikäik. Teisel sisendvõllil asuvad paariskäigud 2, 4 ja 6. Kahe erineva siduri eesmärk on sõidu ajal lülitada sisse järgmine käik ning sellega saavutatakse sujuv käiguvahetus. [2]



Joonis 1.1 Topeltsiduriga käigukast tööpõhimõte [3]

1.2.2 CVT-automaatkäigukast

CVT-automaatkäigukast ehk variaatorkäigukastis toimub ülekanne primaar- ja sekundaarkoonuse vahel tõukelülilindi või plaatveoketiga. Süsteemis on planetaarülekanne, mille külge on ühendatud planetaarmehhanism ja kaks sidurit mehhanismi kontrollimiseks. Kaks erinevat sidurit on edasi- ja tagurpidi liikumiseks. Kui mõlemad sidurid on lahti ühendatud, siis jõuülekannet ei toimu. Kui edaspidi või tagurpidi sidur on ühendatud, veab mootor primaarkoonust ning primaarkoonus oma korda tõukelülilindi või plaatveoketi kaudu sekundaarkoonust. Primaar- ja sekundaarkoonuspaari vahelist ülekandesuhet muudetakse diagonaalselt telgsuunalise nihutamise teel. Väiksem ülekandesuhe saavutatakse kui primaarkoonuspaari lükatakse kokku ning sekundaarkoonuspaar tõmmatakse lahti. Ülekandesuhte muutumist kontrollib käigukasti juhtseade. [2]



Joonis 1.2 Variaatorkäigukasti tööpõhimõte [4]

1.3 Planetaar-automaatkäigukast

Planetaarmehhanismi tüüpi automaatkäigukast on kõige levinum automaatkäigukast tänapäeva autodes. See on väga keeruka ülesehitusega. Käigukastis on kolm peamist komponenti: [2]

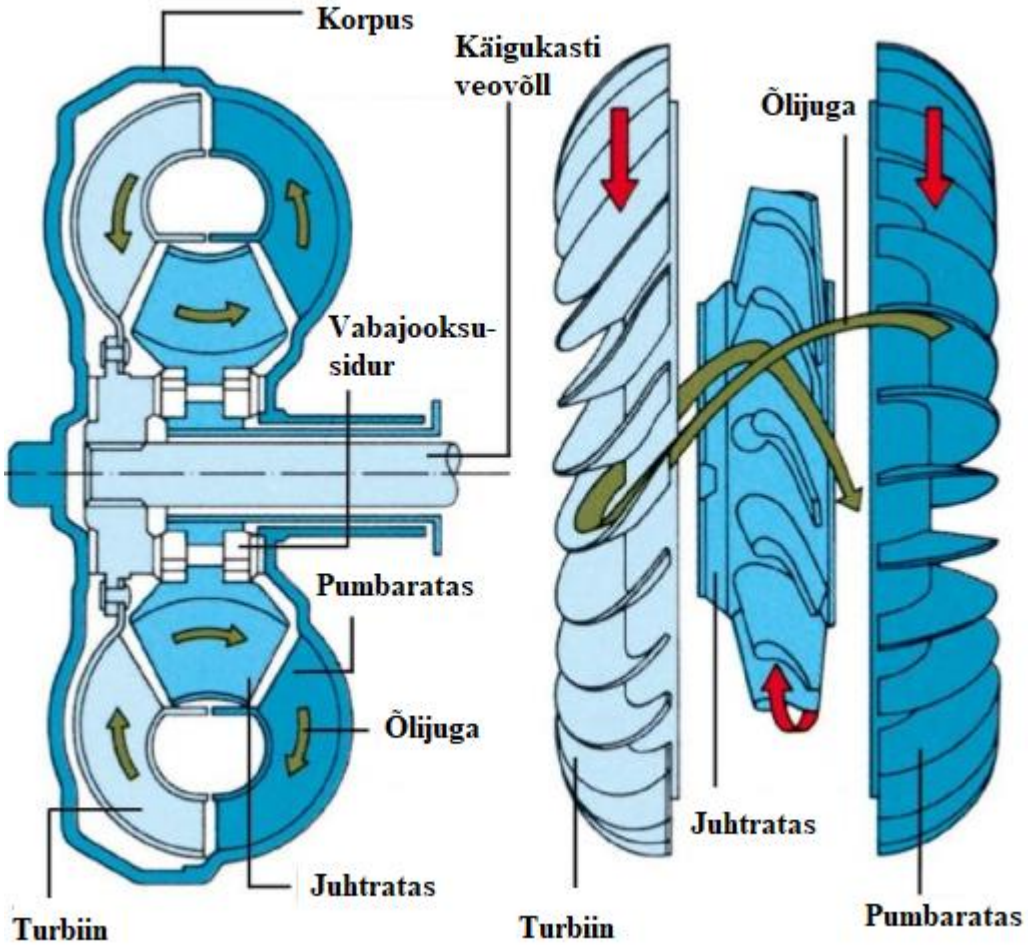
- Hüdrodünaamiline pöördemomendi muundur
- Planetaarmehhanism

- Hüdraulika plokk

1.3.1 Hüdrodünaamiline pöördemomendi muundur

Hüdrodünaamiline pöördemomendi muundur eesmärk on mootori pöördemomendi muundamine ja ülekandmine käigukasti, pehme ja mugava paigaltstardi võimaldamine, mootori väändevibratsiooni summutamine. Hüdrodünaamiline pöördemomendi muundur koosneb pumbarattast, vabajooksuga juhtrattast, turbiinist ja sildamissidurist. Joonis 1.3 on näidatud muunduri tööpõhimõte. Pumbaratas on ühendatud mootori hoorattaga ning selle pöörlemiskiirus on mootori pöörlemiskiirusega sama. Turbiin on ühendatud käigukasti veovõlliga Paigaltstardil seisavad turbiin ja juhtratas ning pumbaratas pöörleb mootori pöörlemiskiirusel. Turbiin alustab pöörlemist kui selle pöördemoment on suurem kui käigukasti veovõlli takistusmoment. Turbiinilt tulev õlijuga voolab pumbarattale pöörlemissuunale vastupidiselt, kuid seda takistab juhtratas, mis vastupidises suunas blokeeritud siduriga ning tänu sellele suunab juhtratas 90° kumerusega õlijuga pumbaratta pöörlemissuunas. See efekt tekitabki pöördemomendi suurenemise ning pöörlemiskiiruste samastumisel langeb õlijuga juhtrattale järjest väiksem nurga all ja vastusurve väheneb. Vastusurve vähenedes väheneb ka pöördemoment pumbarattalt turbiinile. Kui pumbaratta ja turbiini pöörlemiskiiruste vahekord on vahemikus 0,85 kuni 0,9, hakkab õlijuga liikuma otse turbiini labalt pumbaratta labale ning vabastatakse vabajooksusidur ning ühendatakse muunduri sildamissidur ehk ühendatakse omavahel turbiin ja pumbarattas. Sellest hetkest on kasutegur pööretel kuni 97%. Muunduri omadused [2] [5]:

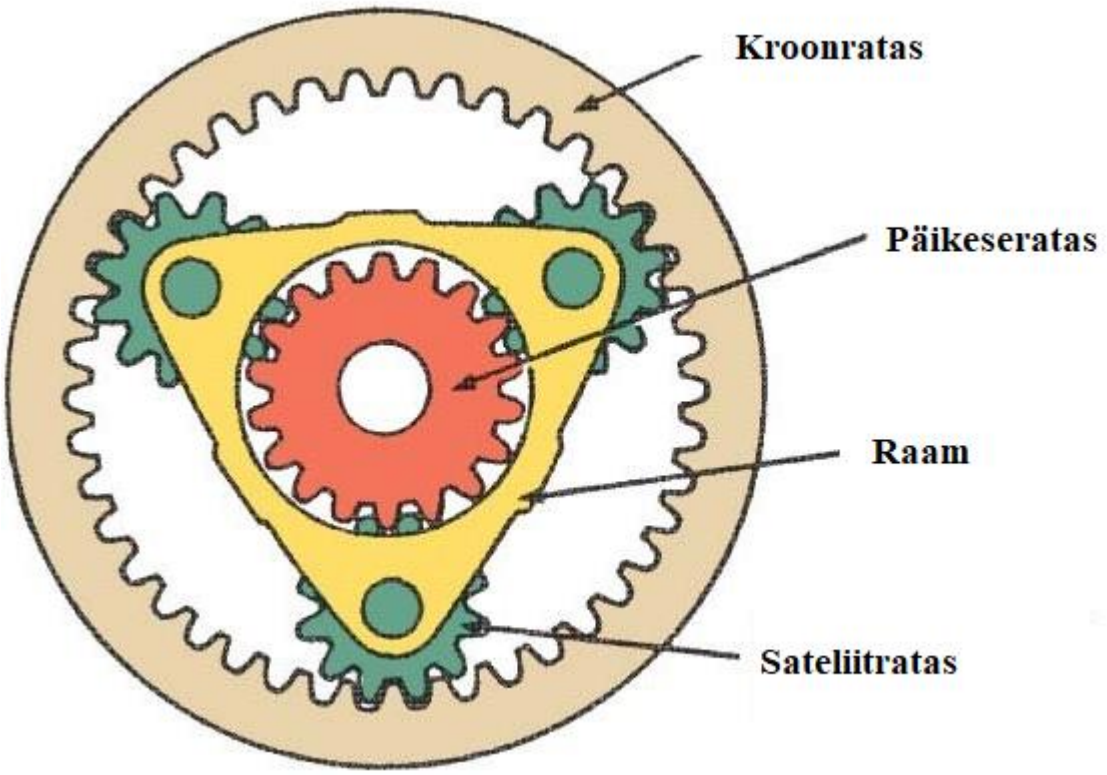
- Puudub mehaaniline kulumine
- Sujuvam paigaltstart
- Mootor ei saa kohalt liikudes „välja surra“
- Pöördemoment võimendus kohandatakse autol automaatselt
- Paigaltstardil on kõige suurem pöördemomendi võimendus



Joonis 1.3 Hüdrodüaamilise pöördemomendi muunduri tööpõhimõte [6]

1.3.2 Planetaarmehhanism

Planetaarmehhanismi eesmärk on tekitada erinevad ülekandesuhted muunduri võlli ja veovõlli vahel. Tavapärane planetaarmehhanism koosneb päikeserattast, sateliitratastest, kroonrattast ja raamist. Joonis 1.4 on näidatud planetaarmehhanism. Sateliitrattad on laagertatud oma telgedega raamile ning need veerevad kroonratta sisehammastel ja päikeseratta välishammastel. Automaatkäigukasti planetaarmehhanismis ei ole kasutusel tavapärane planetaarmehhanism, kuna sellel ei ole piisavalt erinevaid ülekande suhteid. Kombineerides erineva hulga erinevaid päikese-, sateliit-, kroonrattaid või raame saavutatakse piisaval hulgal erinevaid ülekandesuhteid. Erinevad ülekanded saavutatakse päikeseratta, kroonratta või raami seiskamise või vedamisega. Vedavaks osaks planetaarmehhanismis võib olla kroonratas või raam. Planetaarmehhanismi osade kinni hoidmiseks või vabastamiseks kasutatakse sidureid, mida juhib hüdraulika plokk. Planetaarmehhanismi kasutamisel automaatkäigukastis on käiguvahetus aeg väga lühike ning käiku lahutatakse ja lülitatakse samaaegselt. [2] [5]



Joonis 1.4 Planetaarülekanne [7]

1.3.3 Hüdraulika plokk

Hüdraulika ploki eesmärk on juhtida erinevate hüdraulilist süsteemi. Süsteem koosneb õlipumbast, survereguleerklapist, käsivaliku klapist, lülitusklappidest ja reguleerklapist. Hüdraulika plokis juhitakse solenoidklappe, mis omakorda juhivad käigu lülitamiseks hüdraulilisi käiguklappe, mida kasutatakse eelnevas peatükis lahti seletatud planetaarmehhanismi osade sidurdamiseks või lahutamiseks. Selle eesmärk on sujuva käiguvahetuse tagamine. Ploki ülesandeks on ka juhtida peatükis 1.3.1 lahti seletatud muunduri sildamisidurit [2] [5].

2. EELTEGEVUSED

2.1 Ülesande püstitus

Lõputöö eesmärk on välja töötada auto käigukasti mõõteploki põhimõtteline lahendus. Koostatud mõõteplokk koos tarkvaraga peab olema võimeline mõõtma ja salvestama vajalikke andmeid käigukastist vähemalt 1 min etteantud failiformaati. Mõõtmiste tulemusel maksimaalne lubatud häiring on 1%. Mõõtmistele peab saama lisada markereid, et pikast salvestusest leiaks õige käiguvahetuse koha üles. Failis peab olema ka märkmete info lahter, kuhu saab lisada vaheteksti mõõtmise kohta. Seadet kasutatakse autos, kus on saadaval 12 V toitepinge. Käigukasti tüübiks on 09G.

Koostatud mõõteplokk peab olema suuteline mõõtma ja salvestama:

- Nelja solenoidi PWM juhtsignaale
- Käigukasti sisendvõlli pöörlemiskiirus
- Käigukasti väljundvõlli pöörlemiskiirus
- Käigukastiõli temperatuur

Väljundfail peab ühilduma tabelarvutusprogrammiga, kus on võimalik teha järeltöötlust. Fail tuleb ühildada ka *Valve Body* teststendi andmefailiga. Mõõteplokk peab salvestatud faili järgi genereerima identse signaali auto hüdraulikaplokile.

Tegemist on põhimõttelise lahenduse väljaarendamisega, seega ei loeta kõiki vajalike signaale, mida oleks vaja reaalse auto juhtseade imiteerimiseks katsestendis.

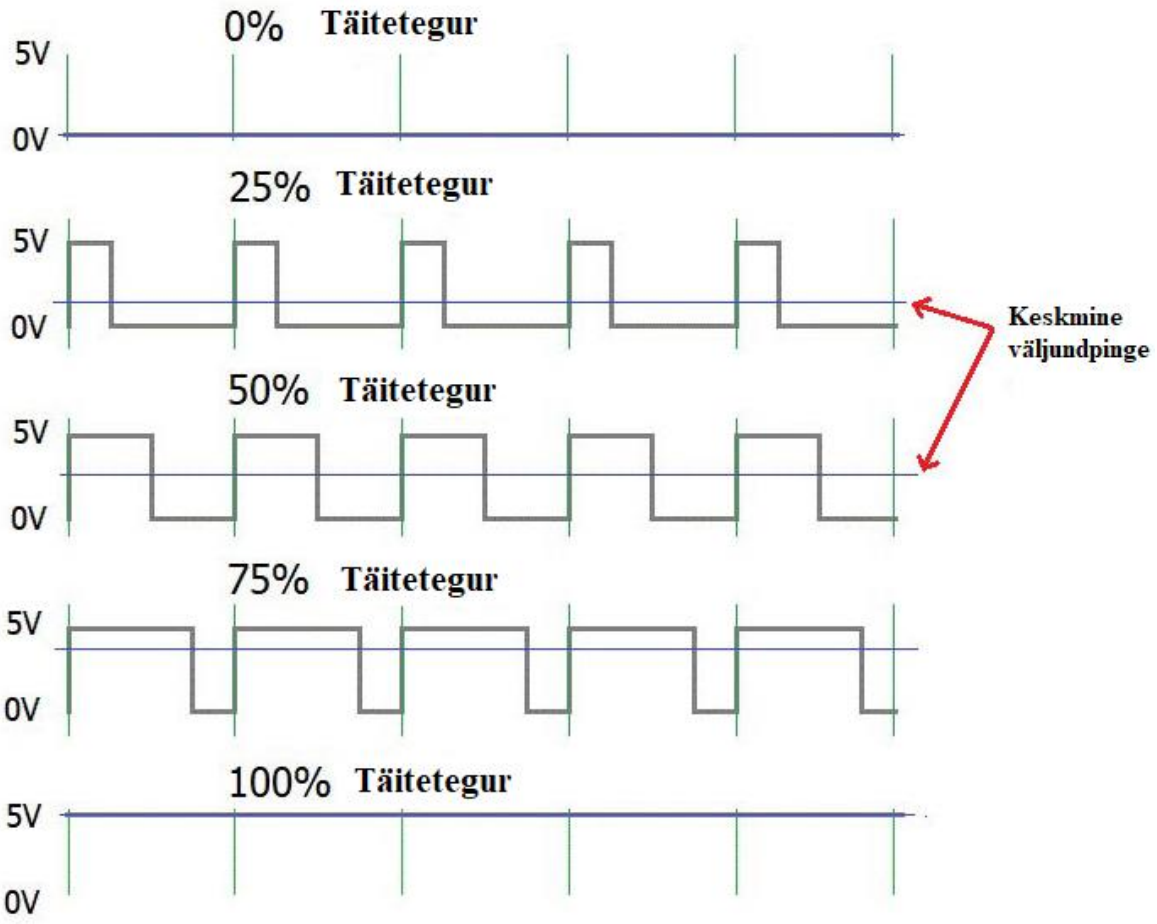
2.2 Signaalide analüüs

Mõõteplokk peab salvestama väga erinevat tüüpi signaale ning iga signaali lugemiseks kasutatakse erinevat meetodikat.

2.2.1 Solenoidi PWM signaal

Solenoidide juhtimiseks kasutatakse PWM(*pulse width modulation*) signaali. PWM signaaliks nimetatakse kastsignaali kujulist lainekuju, millel on kaks väärtust kõrge(1) ja madal(0). Signaali lülitatakse ühtlasel kiirel sagedusel kõrgeks ja madalaks ning muudetakse kõrge signaali laiust.

Signaali periood on pöördvõrdeline signaali sagedusega. Signaali sagedus peab olema piisavalt suur, et koormus ei jõuaks täiesti rakenduda. [8]



Joonis 2.1 PWMi näide [9]

Signaali kõrge oleku laiuse ja signaali perioodi suhet nimetatakse täiteteguriks.

$$D = \frac{PW}{T} \times 100\%$$

Võrrand 1 Täiteteguri arvutamine

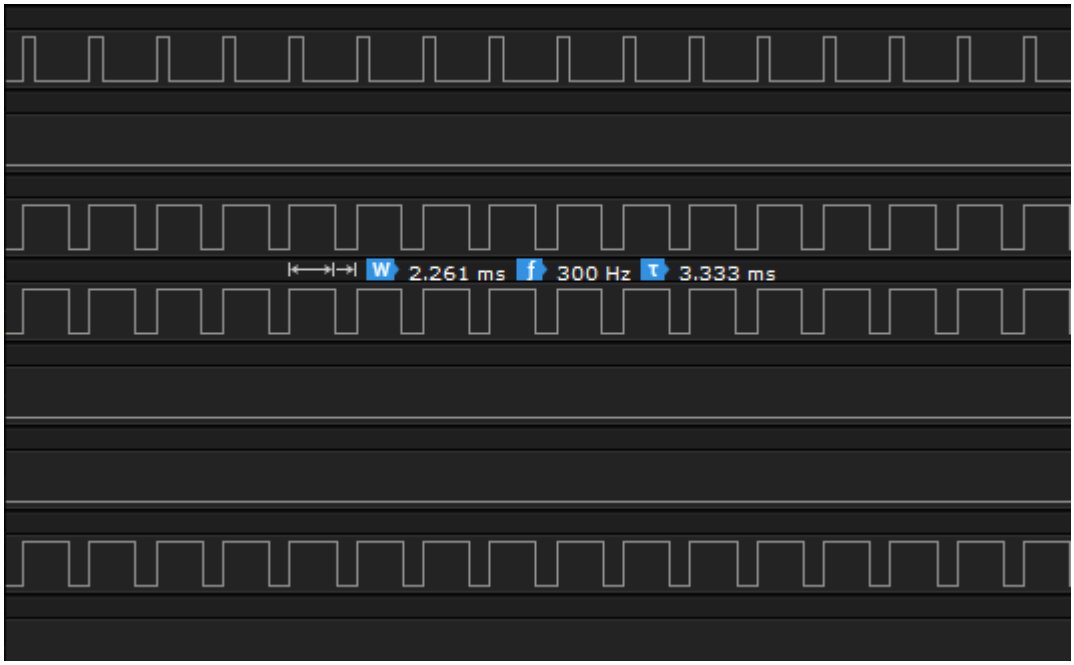
Kus *D* – täitetegur

PW – pulsi laius

T – signaali periood

PWM signaali kasutatakse informatsiooni edastamiseks või võimsuse reguleerimiseks. Solenoidide puhul täiteteguri muutmisega, muudetakse solenoidile kanduvat võimsust. Võimsuse reguleerimisega avaneb või sulgub solenoid.

Mõõteploki teostamiseks oli vaja teada, millise karakteristikuga signaal autos solenoididele jõuab. Selleks teostati autos mõõtmised Saleae Logic Pro 16 loogikaanalüsaatoriga [10].



Joonis 2.2 Solenoidide signaalid

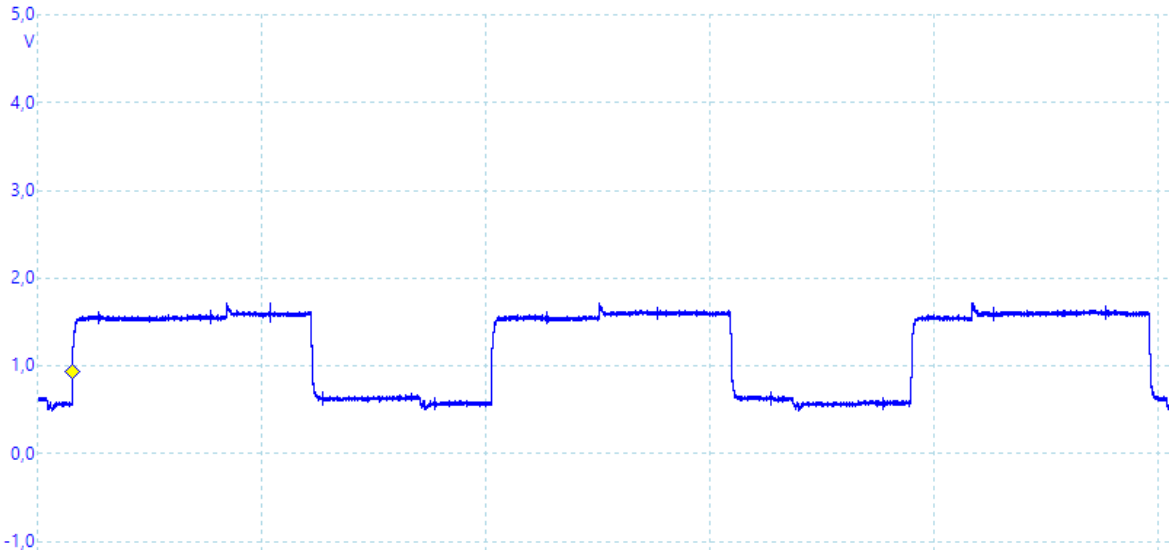
Katse tulemusena leiti, et auto juhtseade juhib PWM signaaliga sagedusel 300 Hz(Joonis 2.2). Signaalide pinge on 12-14 V. Seda tulemust kasutatakse hiljem tarkvara ja elektroonika koostamisel.

2.2.2 Käigukasti sisend- ja väljundkiirus

Käigukastis on sisend- ja väljundkiiruste jaoks kasutusel Hall'i andurid. Hall'i andurid väljastavad PWMiga samasugust kastsignaali (Joonis 2.1). Halli anduri tööpõhimõte seisneb Hall'i efektil. Halli efekt tekitab Halli pinget, kui anduri läheduses on magnet [11]. Halli anduri signaalis muutub nii täitetegur kui ka sagedus. Täitetegur ei ole kiiruse mõõtmise puhul tähtis, kuid sagedus kandubki üle kiiruseks kui mõõta mitu korda sekundis halli andur annab kõrge signaali.

Käigukastis on hammasrataste hammaste küljes magnetid ning igakord kui hammas liigub mööda halli andurist, tekib kastsignaal. Sisend- ja väljundkiirused on vajalikud auto juhtseadmele. Nende kiiruste võrdlemisel saab teada, kas käiguvahetus on toimunud [5].

Ülesande püstitusega lisati ka joonis halli anduri signaalist (Joonis 2.3). Jooniselt selgub, et halli anduri kõrge olek on ligikaudu 1,5 V. Seda teadmist kasutatakse seadme väljaarendamisel.



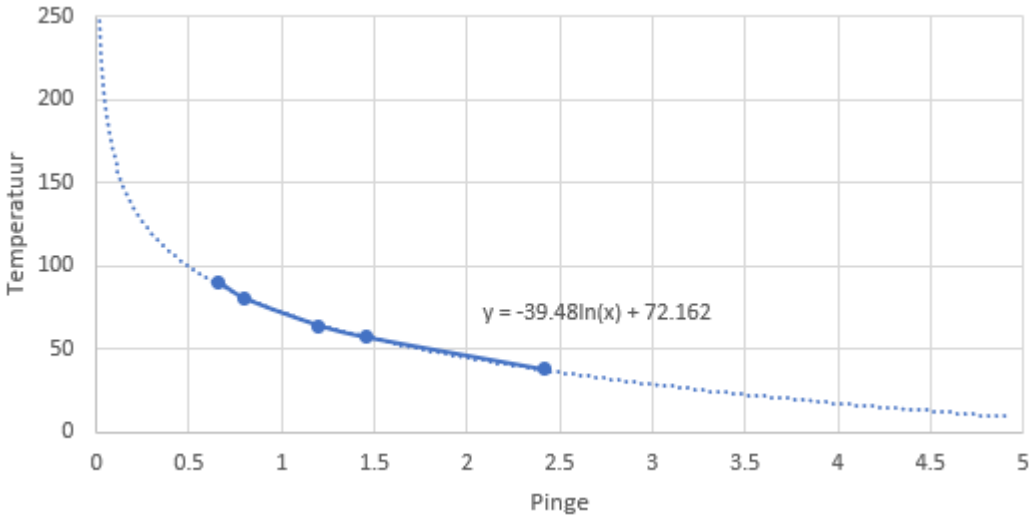
Joonis 2.3 Halli anduri signaal

2.2.3 Käigukastiõli temperatuur

Käigukastiõli temperatuuri mõõtmiseks on käigukastis kasutusel NTC (Negative Temperature Coefficient)-termistor temperatuuriandur. Temperatuurianduri põhimõte seisneb selles, et termistori takistus on sõltuv temperatuurist ning selle takistuse muutumisel muutub pinge kasutades pingejaguri loogikat [12]. Termistori takistus on väga suur kui on madalad temperatuurid ning vastupidi kõrgetel temperatuuridel [5].

Temperatuurianduri väljundpinge jääb alati 0V ja toitepinge vahele. Antud juhul autos on temperatuurianduri toitepingeks 5 V ehk anduri väljund on 0-5 V pinge vahel.

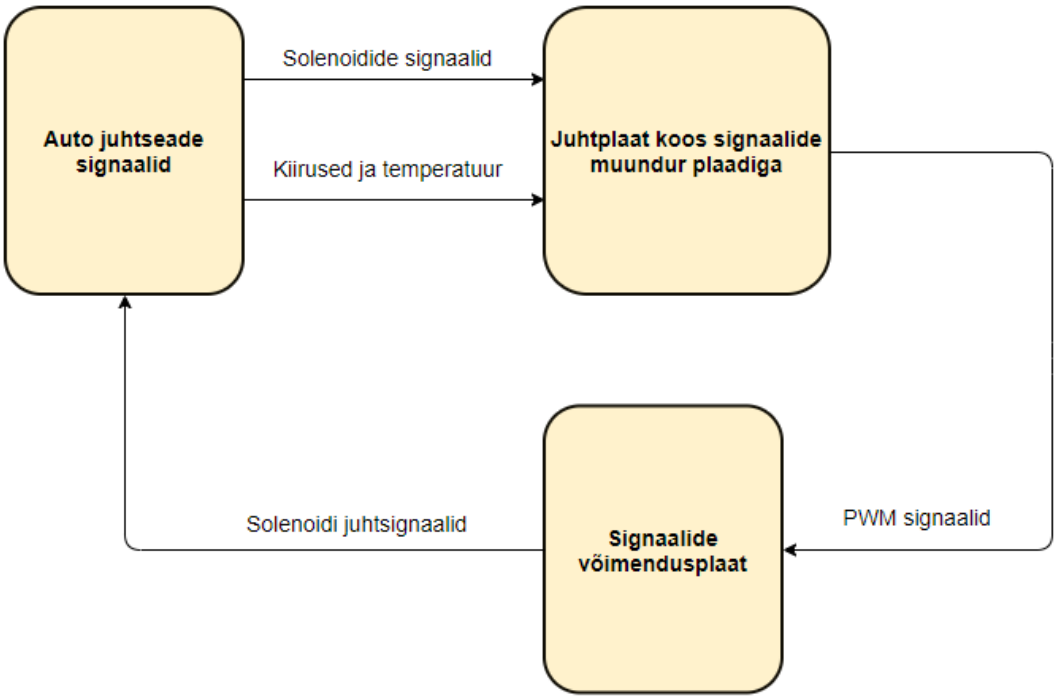
Käigukasti manuaaliga pole kaasa antud täpseid andmeid temperatuurianduri kohta. Ülesande püstitusega lisati kaasa mõningad temperatuuri ja sellele temperatuurile vastavad pinged. Antud andmete puhul loodi graafik, millelt leiti funktsioon (Joonis 2.4). Antud funktsiooni on hiljem tarkvaralises lahenduses vaja temperatuuri arvutamiseks.



Joonis 2.4 Temperatuuri ja pinge sõltuvuse graafik

3. ELEKTROONIKA

Mõõtelahenduse saab jagada kaheks eraldi elektroonika osaks. Esimeseks osaks on mõõteplaat, kuhu jõuavad kõik auto signaalid läbi signaalide muundur plaadi. Teiseks osaks on signaalide võimendus plaat, mis teeb 3,3 V loogikapinge auto solenoididele sobivaks juhtsignaaliks. All on toodud välja skeem süsteemist(**Tõrge! Ei leia viiteallikat.**).



Joonis 3.1 Mõõtelahenduse skeem

3.1 Juhtplaat koos signaalide muundamise plaadiga

Auto toitepinge on 12V ning sellest tulenevalt on enamik signaalidest vaja juhtplaadile sobivaks muundada. Selle jaoks on jaotatud antud peatükk jagatud kaheks erinevaks osaks:

3.1.1 Juhtplaat

Lahenduse väljatöötamiseks oli antud töö autorile STM32F746GDISCOVERY platvorm. Käesoleval platvormil on STM32F746NGH6 mikrokontroller, millel on kasutusel ARM 32bitine Cortex M7 arhitektuur. Mikrokontrolleril on maksimaalne taktsagedus 216MHz, kuni 1MB välkmälu ning 320KB muutmälu. Platvormil küljes on palju erinevaid mooduleid, kuid enamus neist moodulidest ei leia kastutust. Toon alljärgnevalt välja juhtplaadil kasutust leidvad moodulid [13]:

- 4,3 tolline puutetundlik LCD-ekraan, resolutsioon 480x272
- Micro-SD kaardi pesa

4,3 tolline puutetundliku LCD-ekraani on piisavalt suur, et luua sinna kasutajaliides kui ka samaaegselt visualiseerida salvestatavaid signaale. Andmete salvestamiseks kasutatakse micro-SD kaarti ning seetõttu on micro-SD kaardi moodul hädavajalik.

Platvormil on lisaks toodud välja Arduino Uno V3 konnektor, mille järgi sai disainitud muundur plaadi ühendused juhtplaadile. Järgnevas tabelis on välja toodud juhtplaadi ühendused [13]:

Tabel 3.1 Juhtplaadi ühendused

Signaali tüüp	Viigu tähis	Viigu funktsioon
Solenoidi N93 signaal	D6	TIM12_CH1
Solenoidi N283 signaal	D1	TIM8_CH1
Solenoidi N282 signaal	D10	TIM1_CH1
Solenoidi N90 signaal	D9	TIM2_CH1
Sisendkiiruse signaal	D5	TIM5_CH4
Väljundkiiruse signaal	A2	TIM14_CH1
Temperatuurianduri signaal	A0	ADC3_IN0

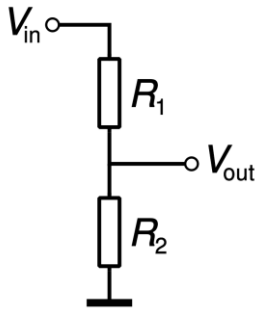
Juhtplaat saab 5V toite läbi auto sigaretisüütaja.

3.1.2 Signaalide muundamise plaat

STM32F746DISCOVERY platvormi toitepinge on 5 V ning enamus plaadi viikudest on 5 V pinge tolerantsed. Seega autost tulevad 12 V signaalid ümber teha 5 V signaalideks. Peatükis 2.2 koostatud signaalide analüüsi järgi on teostatud järgnevad signaalide muundamised.

Solenoide PWM signaalide pinge on 12 V ning need signaalid tuleb vähemalt muundada 5V pinge juurde. Pinge muundamiseks kasutatakse erinevaid eelvalmistatud pingekonvertereid kui ka pingejagurit. Pingejagur on lihtne elektriskeem, mis koosneb kahest takistist. Pingekonverteri ja

pingejaguri peamine vahe on, et pingekonverteriga on võimalik kasutada voolu tarbivaid seadmeid, kuid pingejaguri puhul kuumeneksid takistid suurema voolu korral üle. Kuna signaalidel ei ole voolu, otsustati kasutada pingejagurit.



Joonis 3.2 Pingejaguri skeem

Pingejaguri takistite väärtused arvutati järgneva valemiga [14]:

$$V_{out} = \frac{V_{in} * R_2}{R_1 + R_2}$$

Võrrand 2 Pingejaguri väljundpinge valem

Kus V_{out} – väljundpinge

V_{out} – sisendpinge

R_1 – takistus enne väljundit

R_2 – takistus pärast väljundit

Arvutuste tegemisel tuli arvestada autos juhtseadme poolt tulevate pinge piikidega kuni 24V, ning selle piigi puhul ei tohi signaali pinge ületada 5 V, 12V signaali puhul peab jääma pinge üle 1,8 V, et mikrokontroller tuvastaks kõrge signaali.

Valemi 2 järgi arvutades leiti takistite sobiv kombinatsioon, et täidaks üleval olevaid tingimusi.

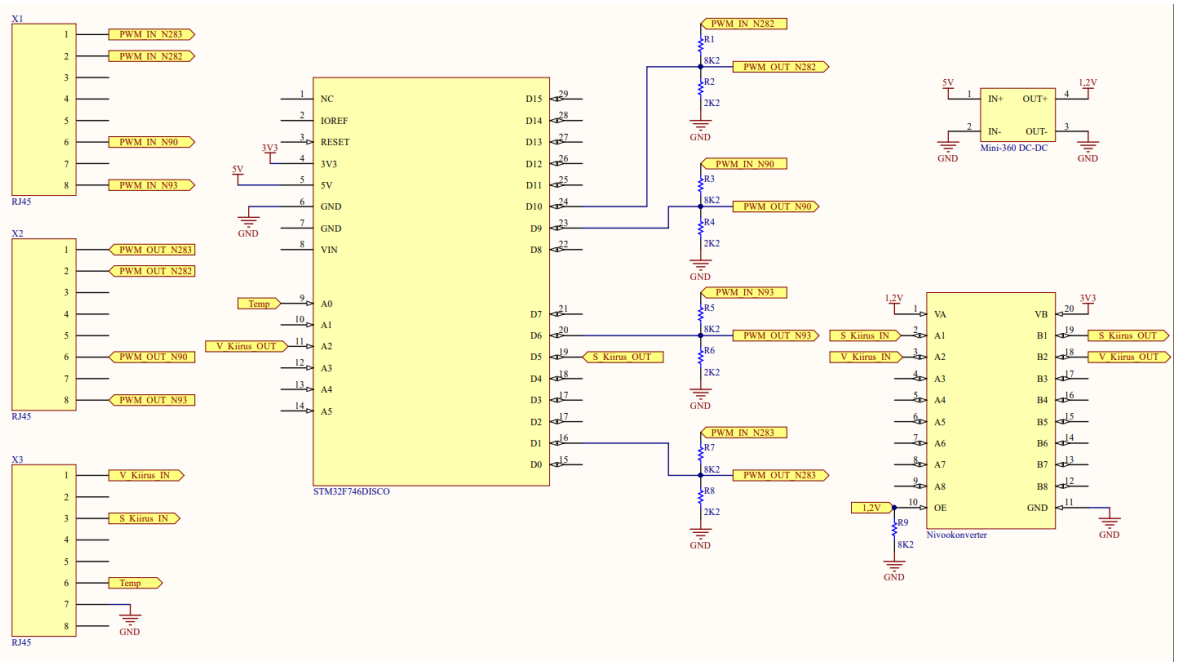
$$V_{out} = \frac{12\ V * 2200\ \Omega}{8200\ \Omega + 2200\ \Omega} = 2{,}538\ V \qquad V_{out} = \frac{24\ V * 2200\ \Omega}{8200\ \Omega + 2200\ \Omega} = 5{,}077\ V$$

Kõigi nelja solenoidi PWM signaali ette pandi elektriskeemis ette pingejagurid sellise konfiguratsiooniga.

Käigukasti sisend- ja väljundkiiruste signaalide pinge on 1,5 V. Neid signaale tuleb võimendada, kuna juhtplaadi jaoks algab kõrge signaal alates pingest 1,8 V. Signaalide pinge suurendamiseks tuli kasutada kahte ostutoodet:

- nivookonverter TXS0108E [15]
- pingeregulaator Mini-360 [16]

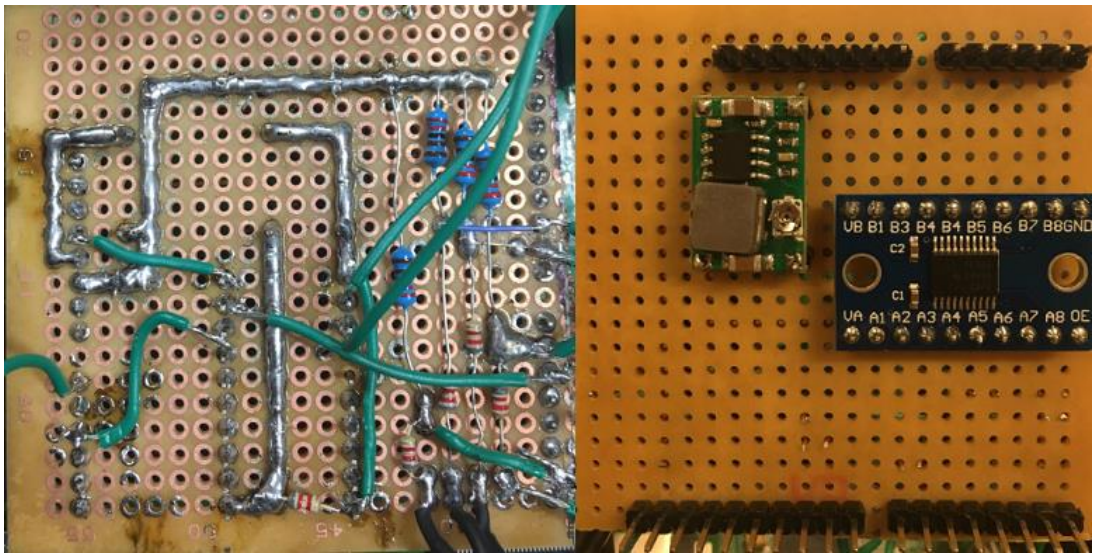
Nivookonverteri eesmärk konverteerida signaali pinge 1,5 V pingele 3,3V. Konverteerimiseks on 1,5 V signaali on konverterile vaja referentspinget 1,5 V. Selle jaoks on kasutusel pingeregulaator Mini-360. Pingeregulaatori sisendiks on juhtplaadilt tulev 5 V pinge ning väljundiks 1,5 V pinge. Mini-360l on eraldi kruvi, millega saab väljundpinget reguleerida. Pinge 1,5 V seati paika kasutades toiteplokki ning multimeetrit.



Joonis 3.3 Juhtplaadi ja signaali muundur plaadi elektriskeem

Elektriskeemi järgides joodeti trükkplaadile skeem, mis on näha Joonis 3.4.

Temperatuuriandurit loetakse otse plaadile kasutades ADC. Analoog-digitaalmuundur teisendab pinge mikrokontrollerile loetavaks väärtuseks.



Joonis 3.4 Muundurplaadi trükkplaat

Kõik ühendused on tehtud RJ45 konnektoriga. Auto solenoidide signaalid jõuavad juhtplaadini X1 konnektorist, kiiruste ja temperatuuri signaalid jõuavad konnektorist X3'st. Võimendusplaadini jõuavad genereeritud signaalid läbi X2 konnektori.

3.2 Signaalide võimendusplaat

Juhtplaadilt genereeritud signaal ei ole võimeline juhtima eraldi käigukasti hüdraulika ploki solenoide. Käigukasti manuaali järgi vajavad solenoidid 12V pinge juures kuni 1A voolu, et juhtida käigukasti tööd. Selle jaoks valmistati eraldi elektriskeem, mis võimendab 3,3 V loogikapinge piisavalt võimsaks signaaliks, et juhtida solenoide. Võimendusplaadi toiteks kasutati 3 elemendiga Li-Po akut.

P-kanaliga Isoleeritud paisuga väljatransistor SPP15P10PLHXKSA1 karakteristikud [17]:

Tabel 3.2 Isoleeritud paisuga väljatransistori karakteristikud

Omadus	Väärtus
Järjepidev vool	15 A
Pinge	100 V

Bipolaartransistor BC635 karakteristikud [18]:

Tabel 3.3 Bipolaartransistori karakteristikud

Omadus	Väärtus
Järjepidev vool	1 A
Kollektori ja emitteri vaheline pinge	45 V
Vooluvõimendus	100 hFE

Schottky diood SB540 karakteristikud [19]:

Tabel 3.4 Schottky dioodi karakteristikud

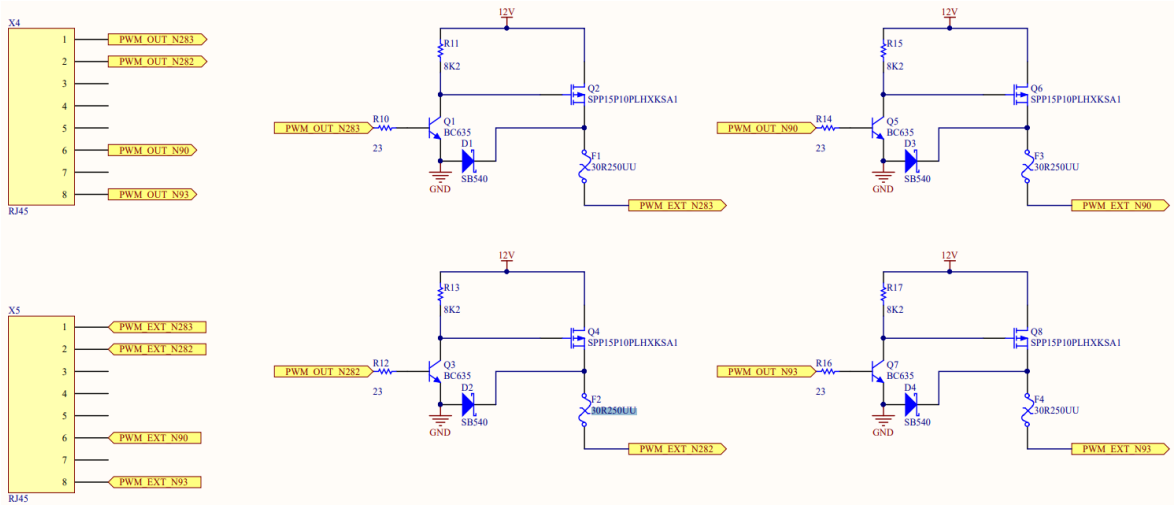
Omadus	Väärtus
Vooluhulk	5 A
Pinge	40 V

Taastuva termokaitsme 30R250UU karakteristikud [20]:

Tabel 3.5 Taastuva termokaitsme karakteristikud

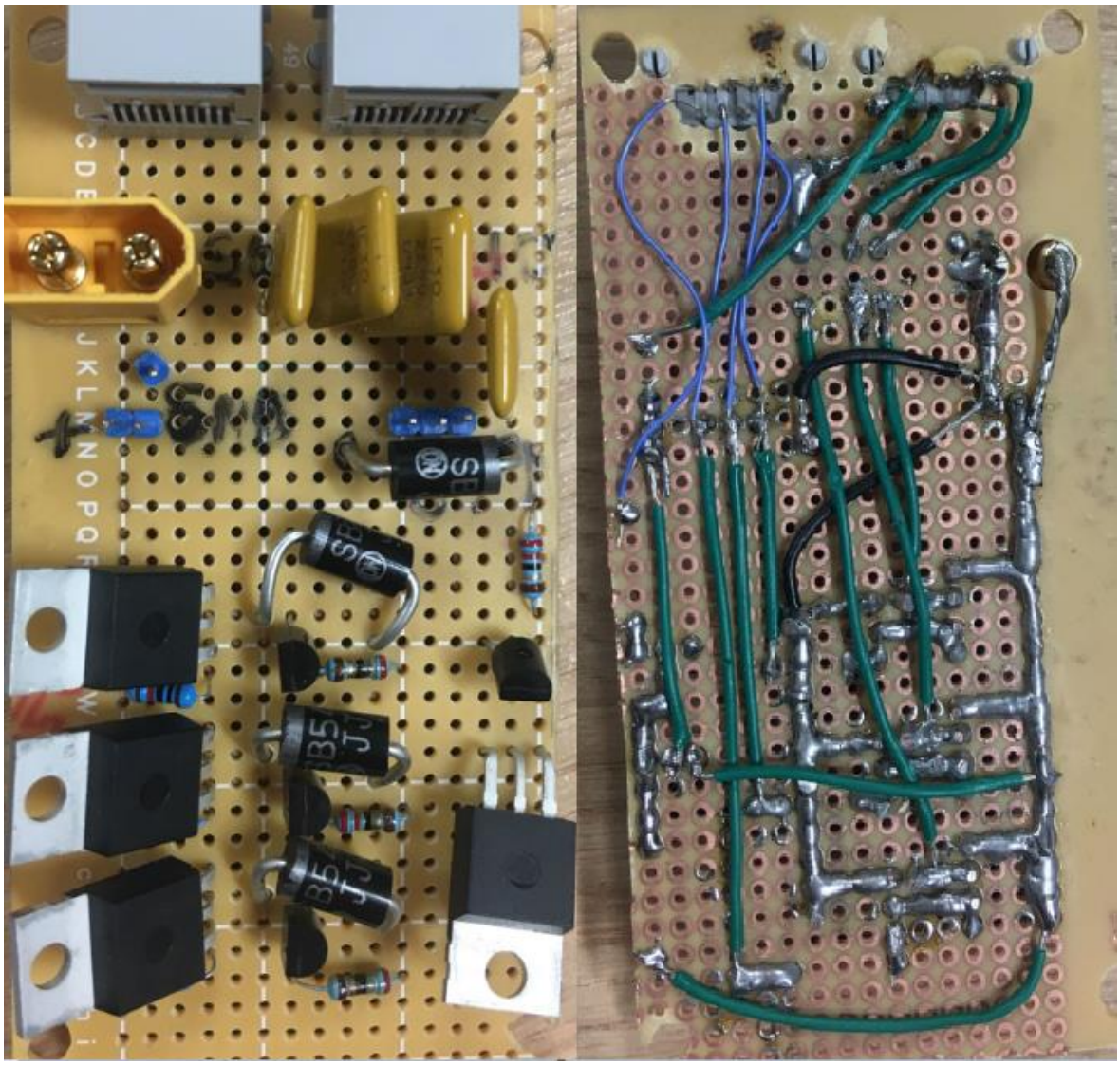
Omadus	Väärtus
Pinge	30 V
Maksimaalne vool enne kaitse aktiveerumist	5 A

Alljärgnevat on näha elektriskeem kokkupandud süsteemist.



Joonis 3.5 Signaalide võimendusplaadi elektriskeem

Elektriskeemi järgi joodeti trükkplaadile kokku võimendusplaadi skeem, mis on näha Joonis 3.6



Joonis 3.6 Võimendusplaadi skeem trükkplaadil

Antud elektriskeemis juhtplaadilt tulev signaal(konnektor X4) juhib BJT transistori baas jalga ning transistor kontrollib p-kanaliga MOSFET'i väravat. *Pull-up* takisti kontrollib, et MOSFET'i värav ei oleks ebamäärases olekus. Schottky dioodi eesmärk on kaitsta MOSFET'i solenoidi lahti lülitamisel induktsioonivoolu eest. Taastuv termokaitse kaitseb süsteemi juhul kui solenoid peaks minema lühisesse. Võimendatud signaal liigub edasi konnektor X5'te.

4. TARKVARA

Lahenduse tegemiseks kasutati STM32CubeMX ja Atollic TrueSTUDIO tarkvarasid. STM32CubeMX tarkvara on graafiline tööriist, millega on võimalik muuta alljärgnevatel konfiguratsioone [21]:

- Viikude konfiguratsiooni
- Väliseadmete ja vahevara konfiguratsiooni
- Mikrokontrolleri taktsageduste konfiguratsiooni

„Atollic TrueSTUDIO“ tarkvara on STM32 mikrokontrolleritele keskendunud integreeritud programmeerimiskeskkond(*IDE*), mis hõlmab endas nii koodi kompilaatorit kui ka koodi silumist [22].

4.1 Programmeerimine

Juhtplaadi programmeerimine teostati C-keeles. Programmide koostamisel kasutati STMicroelectronics näitekoode.

4.1.1 Ülevaade FATFS moodulist

FATFS on mikrokontrolleri ja aplikatsiooni vaheline moodul, mis lihtsustab mikroSD kaardi mooduli kasutamist. Alljärgnevalt on välja toodud põhiliselt kasutud käsud [23]:

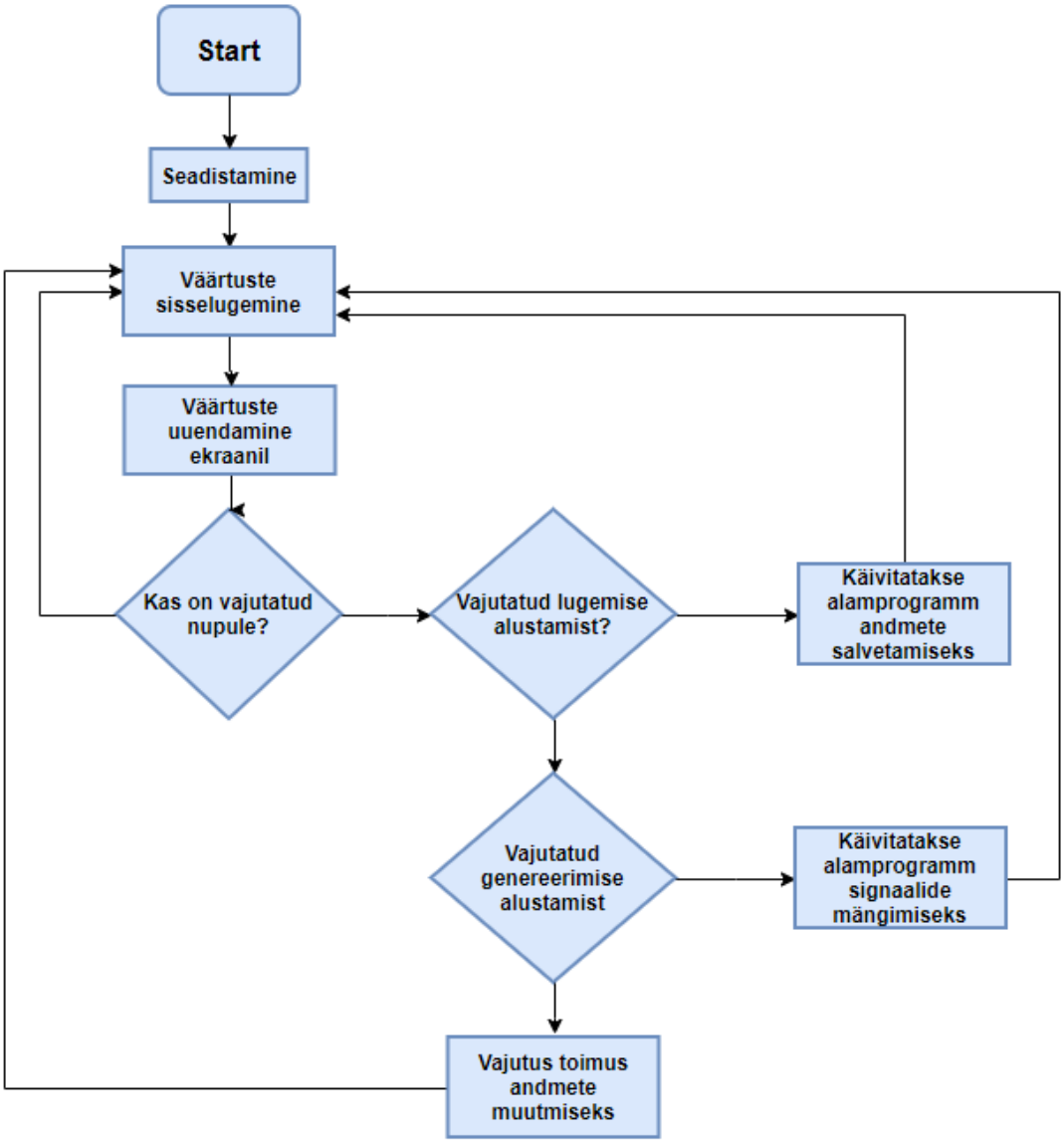
Tabel 4.1 FATFS mooduli käsud

Käsk	Selgitus
FRESULT f_mount (FATFS* fs, const TCHAR* path, BYTE opt);	Loob või katkestab ühenduse failisüsteemi ja mikrokontrolleri vahel. Fs väärtus osutab failisüsteemi objektile. Path väärtus osutab microSD kaardi loogilisele kettale. Opt väärtusest oleneb, kas ühendus luuakse koheselt või oodatakse.
FRESULT f_open (FIL* fp,const TCHAR* path, BYTE mode)	Avab või loob faili failisüsteemil. Fp väärtus osutab faili objekti struktuurile. Path väärtusest avatava või loodava faili nimetus. Mode väärtusega seatakse faili ligipääsetavus.
int f_printf (FIL* fp, const TCHAR* fmt)	Kasutatakse faili kirjutamise eesmärgil. Faili kirjutamine toimub faili algusest. Fp väärtus osutab faili objekti

	struktuurile. Fmtga seatakse paika tekst, mida kirjutatakse faili.
TCHAR* f_gets (TCHAR* buff, int len, FIL* fp)	Funktsiooni eesmärk faili rea lugemine. Failist lugemine algab faili esimeselt realt.
FRESULT f_close (FIL* fp)	Funktsiooni kasutatakse faili sulgemiseks. Fp väärtus osutab faili objekti struktuurile.

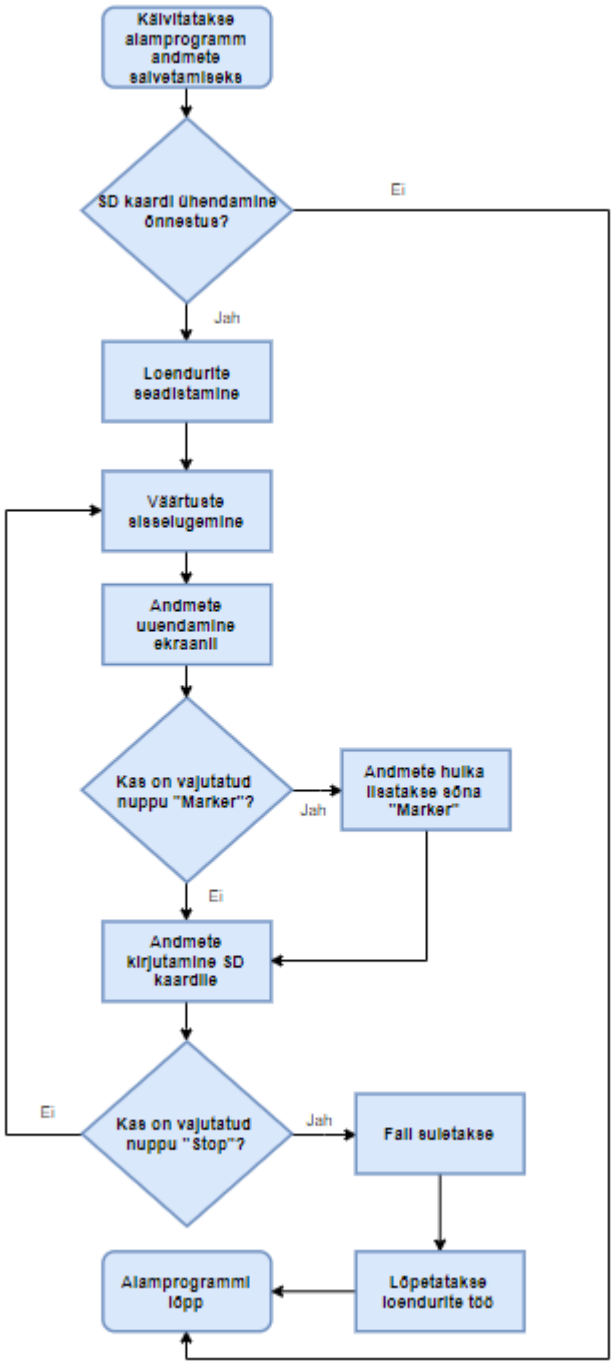
4.1.2 Programmi algoritm

Programm algab erinevate konfiguratsioonide seadistamisega ning kontrollib, kas kõik moodulid töötavad. Kui mõni moodul peaks ebaõnnestuma, läheb programm funktsiooni Error_Handler, mis on lõputu tühi tsükkel. Kui kõik õnnestub ilmub ekraanile kasutajaliides ning programm läheb lõputusse töötsüklisse. Lõputu töötsükli algoritm on näidatud Joonis 4.1 Põhiprogrammi algoritmJoonis 4.1. Kasutajaliides on näidatud LISA 6 Signaali genereerimise alamfunktsioon. Signaalide, mis hõlmavad kastsignaali, lugemine toimub katkestuste baasil. Katkestuste baasil signaalide lugemine toimub kogu aeg kui signaal läheb kõrgeks madalaks või vastupidi. Neid signaale kirjutatakse muutujasse iga kord kui katkestus toimub. Katkestuste kood on välja toodud LISA 3 Katkestuste kood juhtplaadil. Lõputu töötsükkel algab seega temperatuuri signaali lugemisega. Lugemise signaalid muutuvad kasutajaliidesel kogu aeg. Tsükkel kontrollib iga kord, kas ja kuhu on vajutus on tehtud. Kui on tehtud vajutus, käib programm läbi *switch* tingimuse, kus on erinevad kasutajaliidese nupud defineeritud erineva ID'ga. Vastavalt nupu vajutusele toimitakse alamfunktsioonidega edasi, mis on allpool ära kirjeldatud. Fail salvestatakse CSV laiendiga. *CSV* fail on tekstifail, kus salvestatud andmed eraldatakse komadega.



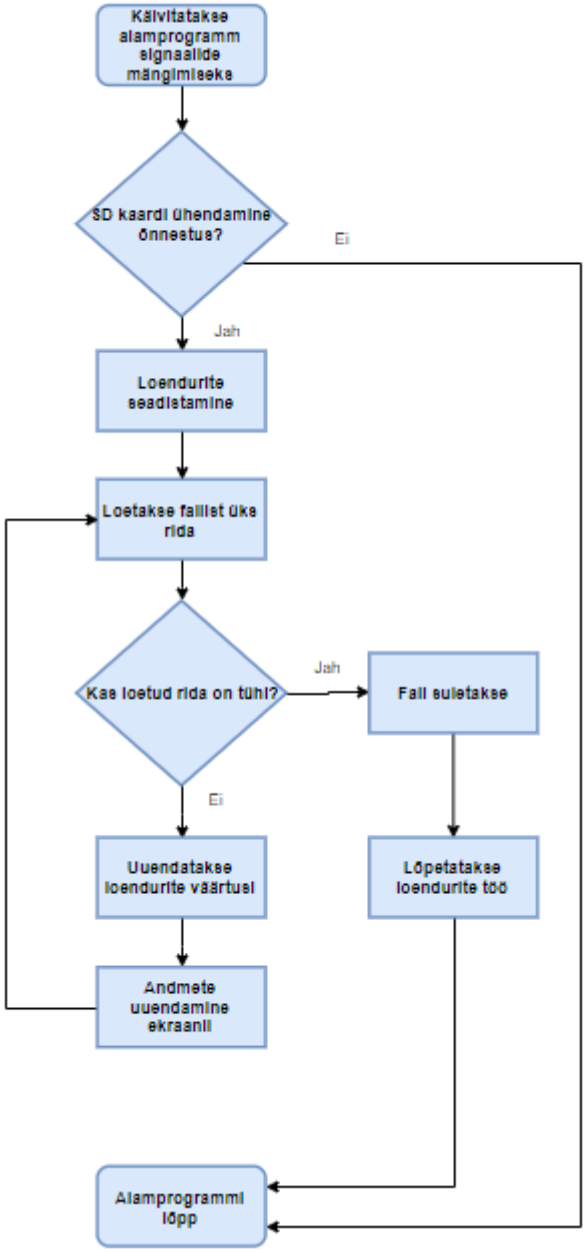
Joonis 4.1 Põhiprogrammi algoritm

Kui on vajutatud faili salvestamise nuppu, käivitub alamfunktsioon Signal_Logging. Alamfunktsioon koodina välja toodud LISA 5 Signaali lugemise alamfunktsioon. Alamfunktsioon loob ühenduse failisüsteemi ja mikrokontrolleri vahel ning seadistab loendurid signaali lugemise režiimi. Kui alamfunktsiooni seadistamised on tehtud, läheb funktsioon tsüklisse, kus signaalide väärtused loetakse sisse, kuvatakse ekraanil ning salvestatakse. Kui vajutatakse salvestamise ajal nuppu „Marker“, kirjutatakse salvestatud andmete juurde ka märge. Alamfunktsioonist väljumiseks on kaustajaliidesel lõpetamise nupp, millele vajutades lõpetatakse salvestamine ja loendurite töö ning katkestatakse ühendus failisüsteemi ja mikrokontrolleri vahel. Alamfunktsiooni algoritm on välja toodud Joonis 4.2.



Joonis 4.2 Andmete salvestamise alamprogrammi algoritm

Kui on vajutatud failist lugemise nuppu, käivitub alamfunktsioon Signal_play. Alamfunktsioon on välja toodud LISA 6 Signaali genereerimise alamfunktsioon. Alamfunktsioon loob ühenduse failisüsteemi ja mikrokontrolleri vahel ning seadistab loendurid signaali genereerimise režiimi. Kui alamfunktsiooni seadistamine on tehtud, läheb funktsioon tsüklisse, kus signaalide väärtused loetakse failist sisse, kantakse ekraanile ja seadistatakse loendurite PWM signaali täitetegur. Alamfunktsioon lõppeb kui failist loetakse tühi rida või vajutatakse kasutajaliidesel lõpetamise nuppu.



Joonis 4.3 Signaalide mängimise alamprogrammi algoritm

4.2 Genereeritud faili ühildamine Valve Body teststendiga

Salvestatud fail tuleb salvestada konverteerida CSV laiendiga failist TSC laiendiga failiks. TSC laiendiga faili ülesehitus on põhineb XML failil. Genereeritud faili ühildamiseks kasutatakse Python programmeerimiskeelt ning kasutatakse IDLE IDE't [24]. Ülesande püstituses oli kaasa antud Valve Body teststendi juhtseade failinäidis. Faili uurides leiti, et suur osa andmetest jäävad alati samaks, muutub ainult solenoidide juhtsignaalide väärtused. Andmete osa, mis jääb samaks, kirjutatakse

otse faili XML struktuuri järgides. Salvestatud failis ignoreeritakse esimest rida ja järgnevalt võetakse iga rea lugem, sorditakse välja solenoidide juhtsignaalid ning kirjutatakse need TSC laiendiga failis õigesse kohta (Joonis 4.4).

```
next(reader)
for row in islice(reader, 1, num_lines - 2):
    f.write('    <TestScriptFrame>\n')
    f.write('      <FrameIndex>'+ '{}'.format(i) +'</FrameIndex>
    f.write('      <Duration>150</Duration>\n')
    f.write('      <UserPrompt></UserPrompt>\n')
    f.write('      <UserPromptEnabled>true</UserPromptEnabled>\n
    f.write('      <ChannelDrivesArray>\n')
    f.write('        <int>0</int>\n')
    f.write('        <int>0</int>\n')
    f.write('        <int>' + row[1] + '</int>\n')
    f.write('        <int>' + row[0] + '</int>\n')
    f.write('        <int>' + row[2] + '</int>\n')
    f.write('        <int>0</int>\n')
    f.write('        <int>0</int>\n')
    f.write('        <int>' + row[3] + '</int>\n')
    f.write('        <int>0</int>\n')
    f.write('      </ChannelDrivesArray>\n')
    f.write('      <PressureReadValuesArr>\n')
    f.write('        <float>0</float>\n')
    f.write('        <float>0</float>\n')
    f.write('        <float>0</float>\n')
    f.write('        <float>0</float>\n')
    f.write('        <float>0</float>\n')
    f.write('        <float>0</float>\n')
    f.write('        <float>0</float>\n')
    f.write('        <float>0</float>\n')
    f.write('        <float>0</float>\n')
    f.write('        <float>0</float>\n')
    f.write('        <float>0</float>\n')
    f.write('        <float>0</float>\n')
    f.write('        <float>0</float>\n')
    f.write('        <float>0</float>\n')
    f.write('      </PressureReadValuesArr>\n')
    f.write('      <CurrentReadValuesArr>\n')
```

Joonis 4.4 Salvestatud faili konverteerimine

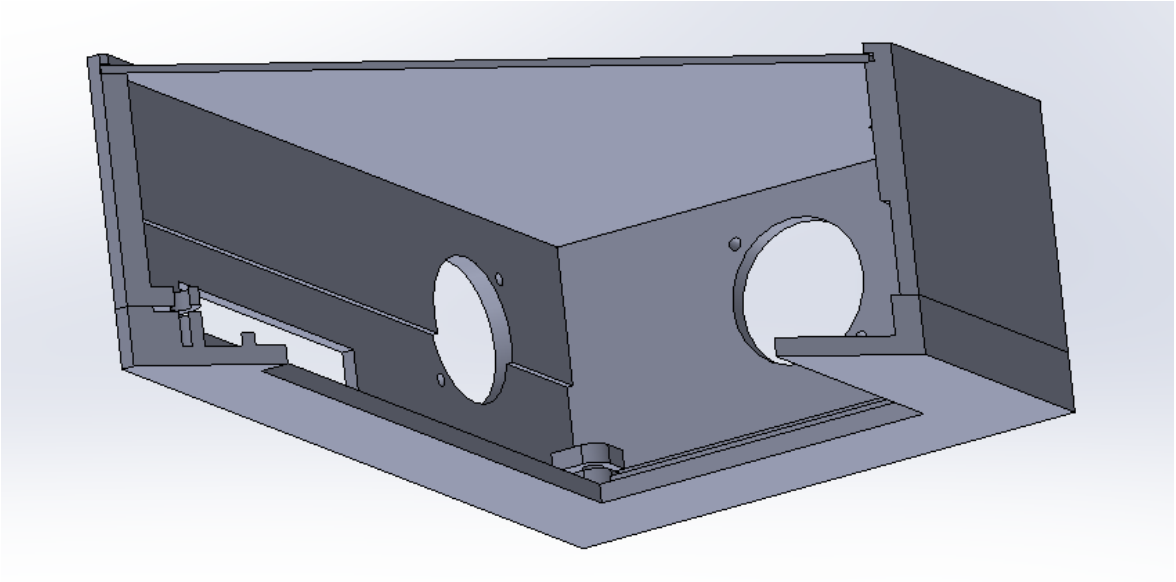
5. MEHAANIKA

Mõlemale joodetud trükkplaadile tehti korpus. Korpuse modelleerimiseks kasutati programmi Solidworks. Korpuse tegemisel lähtuti asjaolust, et kõik trükkplaadid mahuksid sisse ning oleks võimalikud minimaalsed. Korpused prinditi välja TTÜ Robotiklubi 3D-printeriga.

5.1 Juhtplaadi ja signaali muunduri korpus

Juhtplaadi korpuse eesmärk on terve süsteemi hoiustamine koostati kolmest osast

- Alumine osa, mis hoiab juhtplaati ning sellest oli väljalõigatud LCD-ekraani suurune auk.
- Keskmine osa, mis kinnitub kruvidega läbi juhtplaadis olevate aukude alumise osa külge. Keskmise osa küljes oli ka kaks RJ45 konnektorit. Keskmisest osast oli välja lõigatud kaks auku, et oleks ligipääsetavus micro-SD kaardile ning USB aukudele.
- Pealmine osa, mis kattis terve elektroonika. Pealmine osa tehti keskmisesse ossa sisse libisevaks, et oleks elektroonikale kiire ligipääsetavus ning sellega hoiti kinni ka ühte RJ45 konnektorit.



Joonis 5.1 Juhtplaadi korpuse lõige

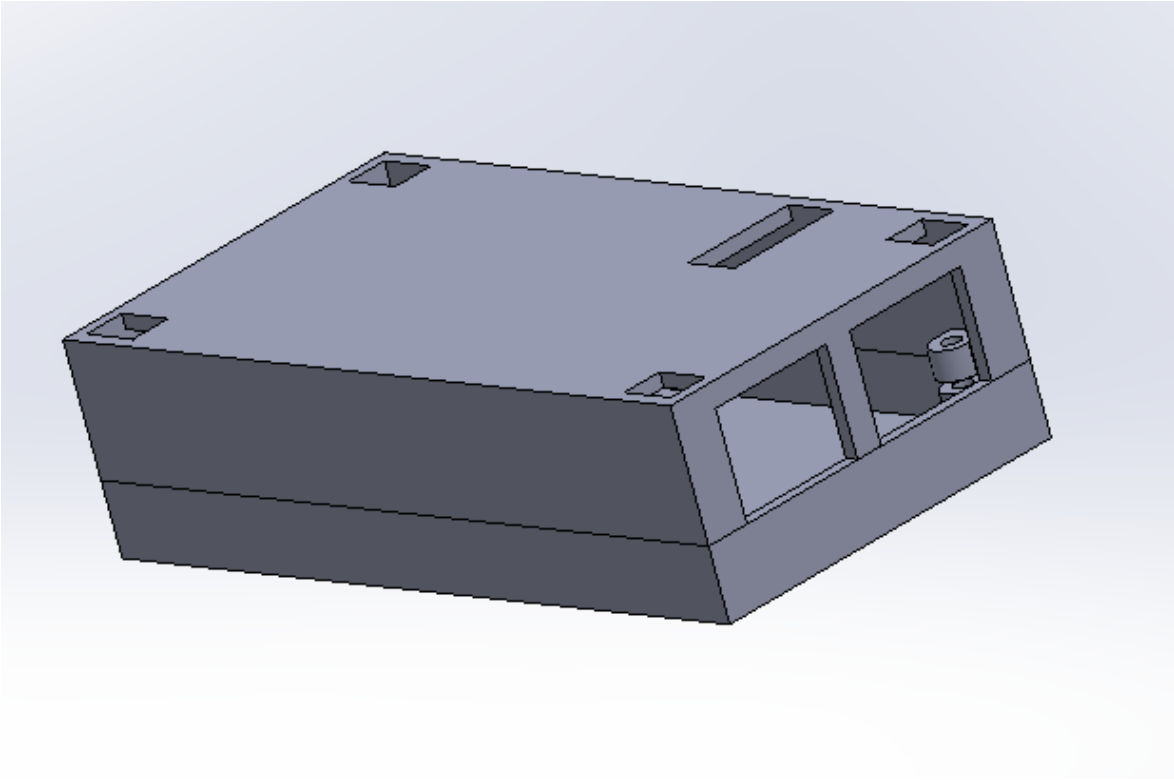
3D-prinditud korpus on näha D

LISA 1 Pildid juhtplaadist ning kasutajaliidesest.

5.2 Võimendusplaadi korpus

Võimendusplaadi korpus koostati kahest osast:

- Alumine osa, kuhu kinnitub joodetud trükkplaati
- Pealmine osa, kust on välja lõigatud kaks RJ45 konnektori suurust auku, XT-60 Li-Po konnektori suurune auk ning neli auku kõikides nurkades, et oleks võimalik kruvikeerajaga kruvidele ligi pääseda ning kaks plaati omavahel kokku panna.



Joonis 0.1 Võimendusplaadi korpus

3D-prinditud korpus on näha LISA 2 Pilt võimendusplaadist.

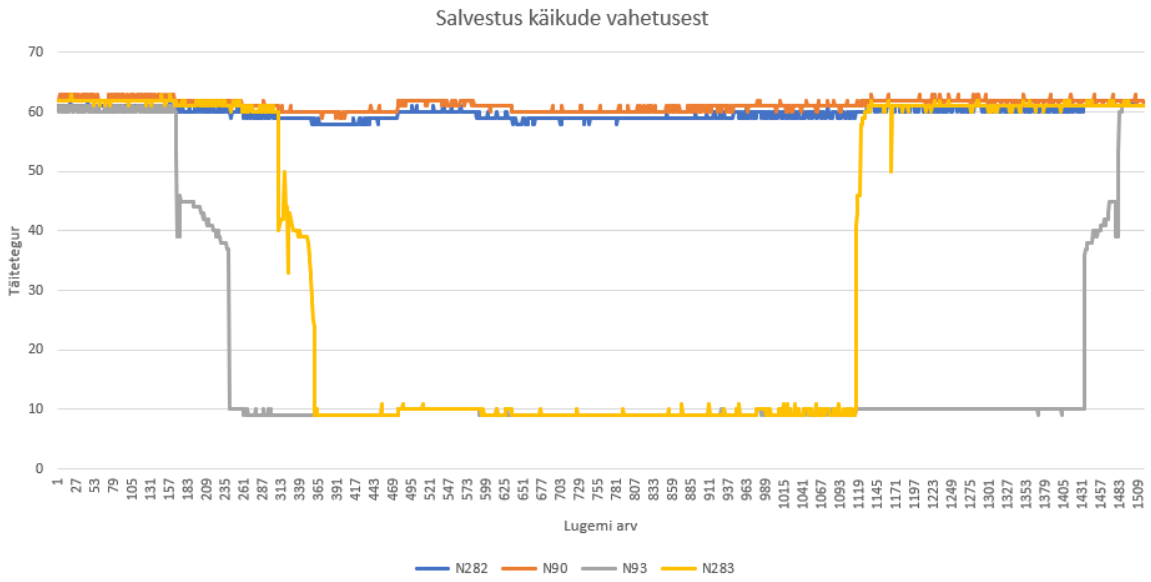
6. KATSED

6.1 Katsed laboritingimustes

Laboritingimustes testiti kõiki elektriskeeme ning tarkvara lahendusi. PWMi signaalide lugemiseks kasutati AFG-2005 funktsioonigeneraatorit [25] ning simuleeriti 300Hz PWMi signaali lugemist ja salvestamist. Katsetused olid edukad. Võimendusplaadi testimiseks kasutati 12V harjastega alalisvoolu mootoreid ning testiti testfailiga. Testfailiks oli fail, kus kõik nelja kanali täitetegureid viid 0-100ni ja tagasi ning mootorid toimisid täpselt nagu pidid. Temperatuuri anduri signaali testimiseks kasutati lihtsat potentsiomeetri skeemi, kus üks muutuv pinge oli anduri signaali sisendiks. Anduri signaali lugemine töötas. Lisaks lasti tervel plaadil salvestada ligikaudu 5 minutit pikk salvestus, et testida, kas on võimalik salvestada minuti pikkune salvestus. Juhtplaat oli võimeline salvestama 5 minuti pikkuse salvestuse.

6.2 Katsed autol

Mõõteploki põhimõttelist lahendust katsetati 10.05.2019 Volkswagen Passati peal. Auto automaatkäigukastist tulevaid solenoidide signaale oli võimalik ekraanilt jälgida, kuid esines palju signaalide häireid. Faili salvestamisega oli ka probleeme, paljudel juhtudel pärast katset ei olnud faili tekkinud või oli faili salvestus poole katse pealt ära lõppenud. Saadi üks korralik salvestus, kus eemaldati käsitsi signaalide häiringud. Joonis 6.1 on näha auto reaalsed käiguvahetused. Graafikul on näha kolme erinevat käiku. Salvestus algab neutraal käigult ning salvestuse käigus auto vahetab neutraalkäigult esimesele, esimeselt teisele, teiselt esimesele ja tagasi esimeselt neutraalkäigule.Joonis 6.1 Solenoide PWM signaalid käikude vahetamisel



Joonis 6.1 Solenoide PWM signaalid käikude vahetamisel

Salvestust prooviti ka ette mänginda läbi võimendusplaadi auto automaatkäigukastile, kuid esines probleem automaatkäigukasti juhtseadme välja lülitamisega ehk käigukasti juhtseade kontrollis ikkagi solenoide. Sellest veast saadi alles aru kui võimendusplaadi paar komponenti põlesid läbi tagurpidi voolu pärast. *Valve Body* teststendile ühildatud faili ei katsetatud.

6.2.1 Katse tulemused

Katsete tulemusena saadi aru, et põhimõtteline lahendus oli enamvähem edukas, kuid tuleb veel arendada tarkvaraliselt ning elektrooniliselt. Juhtplaat tuleb eraldada auto toitest ning luua juhtplaadile oma toiteallikas. Auto signaalide lugemisel kasutada optilist sidestit, et eemaldada auto toitest tulenevad häiringud juhtplaadis. Tarkvaraliselt tuleks edasi liikuda operatsioonisüsteemi kasutamisele, et samal ajal töötavad asjad üksteist ei segaks.

KOKKUVÕTE

Lõputöö eesmärgiks oli välja arendada mõõteploki põhimõtteline lahendus automaatkäigukasti signaalide salvestamiseks. Mõõteploki põhimõtteliseks lahenduseks oli autorile ette antud platvorm, millel tuli lahendus üles ehitada. Koostatud süsteem pidi olema võimeline lugema ja salvestama 4 automaatkäigukasti hüdraulika ploki solenoidide signaali, käigukasti sisend- ja väljundkiiruseid ja käigukastiõli temperatuuri. Kõiki signaale tuli lugeda Volkswagen Passat'i käigukastist, mille tüübiks oli 09G.

Lõputööd alustati automaatkäigukasti lahti seletamisest, et oleks arusaadav, mis signaale salvestatakse. Salvestatavatest signaalidest teostati analüüs mõistmaks, kuidas tuleb läheneda lahenduse leidmisel.

Mõõtelahenduse väljaarendamisel jaotati lahendus kaheks – sisendsignaalid ja väljundsignaalid. Sisendsignaalide jaoks koostati elektriskeem pingejaguritest, nivookonverterist ja pingeregulaatorist. Elektriskeemi eesmärk oli teha autost tulevad signaalid juhtplaadile mõõdetavaks signaaliks. Autole signaalide tagasimängimiseks koostati väljundsignaalide elektriskeem, mille eesmärk juhtplaadilt tulev signaal võimendada hüdraulika ploki solenoide juhtivaks signaaliks. Elektriskeemid koostati kasutades Altium Designer tarkvara. Mõlemale mõõtelahenduse osale loodi programmis Solidworks korpused, mis hiljem 3D-prinditi välja.

Vastavalt ülesande püstitusele loodi juhtplaadile mikrokontrolleri programm. Tarkvara arendamine toimus C keeles ning hõlmas endas kasutajaliidese programmeerimist, erinevate signaalide lugemist ning micro-SD kaardi mooduli juhtimist. LCD-ekraanil näidatakse reaalajas signaale ning on võimalik valida signaalide salvestamise või maha mängimise vahel. Signaalide salvestamisel loodi uus fail, kuhu kirjutati kõik vajalikud andmed. Hiljem kasutati oli võimalik sama faili lugeda ning lugemisest mängida tagasi identset signaali. Salvestatud faili formaat ühildub tabelarvutusprogrammiga. Mikrokontrolleri programmeerimiseks kasutati STM32CubeMX ja Atollic TrueStudio tarkvara.

Laboritingimustes toimunud katsed olid kõik edukad, kuid reaalses katsetustes oli puudusi ning osa katseid ei olnud võimalik teostada tänu nende puudustele.

Põhimõttelise lahenduse edasiarendamiseks oleks vaja muuta elektroonilist lahendust ning juhtplaadi tarkvara ehitada üles operatsioonisüsteemile.

SUMMARY

The aim of the thesis was to develop a measuring device for measuring and logging automatic transmission signals. To develop a solution for the measuring device the author was given a platform, on which, the solution had to be built on. The assembled system was supposed to be able to read and log 4 solenoid signals of the hydraulic block, gearbox input and output speeds and the temperatue of gearbox oil. All the signals had be read from the car Volkswagen Passat, which had 09G type gearbox.

The thesis begins by explaining the fundamentals of automatic transmission to understand what signals are being recorded. An analysis of the signals was made for the purpose of understanding how to approach the solution.

In the development of the measuring device, the solution was dividend into two – input signals and output signals. For the input signals, an circuit diagram was drawn, consisting of voltage dividers, level converter and voltage regulator. The purpose of the scheme was to make the signals from the car readable to the control board. To playback the signals for the car, another circuit diagram was drawn. The purpose of the scheme was to amplify the signals from the control board to the solenoids of the hydraulic block. The circuit diagrams were drawn in Altium Designer software. For both parts of the measurment solution an housing was made by using Soldiworks software. The housings were later 3D printed.

According to the task, a microcontroller programm was created for the control board. The software was developed using C language and it included programming user interface, reading various signals and controlling the micro-SD card module. On the LCD screen real-time signals were displayed and the user could choose between logging and playing back the signals. When logging the signals, a new file was created, where all the necessary data was written. Later it was possible to read the same file and play back the identical signals from the file. The saved file is compatible with the spreadsheet program. The microcontroller was programmed using STM32CubeMX and ATollic TrueStudio software.

All the testing in laboratory conditions were successful, however there were shortcoming in the real tests and some test could not be perfomed because of the shortcomings.

For further development of the system, it is necessary to change the electronics solution and the control board’s software should be built on an operating system.

7. Kasutatud kirjandus

[1] „Tutvustus: Valve body katsestend," SuperFlow Dynamometers & Flowbenches, [Võrgumaterjal]. Saadaval: https://www.superflow.com/aspx/prodDetail.aspx?prodid=26&catid=6&navid=12. [Kasutatud 18 mai 2019].

[2] V. Europa-Lehrmittel, Autonduse käsiraamat, Autoerialade Kirjandus OÜ, 2014.

[3] „Pilt: Topeltsiduriga käigkukast," [Võrgumaterjal]. Saadaval: https://en.wikipedia.org/wiki/Dual-clutch_transmission. [Kasutatud 19 Mai 2019].

[4] B. Bonsen, R. J. Pulles, S. W. Simsons, M. Steinbuch ja P. Veenhuizen, „Implementation of a slip controlled CVT in a production vehicle," 2005.

[5] Automatic Transmission Service Group, „Manuaal: 09G käigukast," 2010. [Võrgumaterjal]. Saadaval: https://shop.ukrtrans.biz/wp-content/uploads/catalogs/09G.pdf. [Kasutatud 18 mai 2019].

[6] Konradin Medien GmbH, „Pilt: Hüdrodünaamilise pöördemomendi muundur," [Võrgumaterjal]. Saadaval: https://www.wissen.de/lexikon/kennungswandler. [Kasutatud 19 mai 2019].

[7] P. Mishra, „Pilt: Planetaarülekanne," [Võrgumaterjal]. Saadaval: https://www.mechanicalbooster.com/2017/12/epicyclic-gearbox-main-components-working-application.html. [Kasutatud 19 mai 2019].

[8] P. Horowitz ja W. Hill, The Art of Electronics, New York: Cambridge University Press, 2015.

[9] CircuitDigest, „Pilt: Pwm signaal," [Võrgumaterjal]. Saadaval: https://circuitdigest.com/tutorial/what-is-pwm-pulse-width-modulation. [Kasutatud 19 mai 2019].

[10] „Manuaal: Saleae Logic Pro 16," Saleae Inc, [Võrgumaterjal]. Saadaval: http://downloads.saleae.com/specs/Logic+Pro+16+Data+Sheet.pdf. [Kasutatud 18 Mai 2019].

[11] J. S. R. V. D. a. O. V. D. B. Pengra, „Tutvustus: Halli efekt,“ 19 juuni 2015. [Võrgumaterjal]. Saadaval: http://courses.washington.edu/phys431/hall_effect/hall_effect.pdf. [Kasutatud 18 mai 2019].

[12] A. Rahman, „Assignment on Temperature Sensors,“ 2018.

[13] STMMicroelectroniks, „Andmeleht: STM32F746DISCOVERY,“ 2017. [Võrgumaterjal]. Saadaval: https://www.st.com/content/ccc/resource/technical/document/user_manual/f0/14/c1/b9/95/6d/40/4d/DM00190424.pdf/files/DM00190424.pdf/jcr:content/translations/en.DM00190424.pdf. [Kasutatud 18 mai 2019].

[14] „Tutvustus: Pingejagur,“ [Võrgumaterjal]. Saadaval: https://sisu.ut.ee/elektroonika/26-pingejagur-potentsiomeeter-reostaat. [Kasutatud 19 mai 2019].

[15] Texas Instrument, „Andmeleht: TXS0108E nivookonverter,“ detsember 2007. [Võrgumaterjal]. Saadaval: https://www.elecrow.com/download/txs0108e.pdf. [Kasutatud 19 mai 2019].

[16] Monolithic Power Systems, „Andmeleht: Pingeregulaator Mini-360,“ [Võrgumaterjal]. Saadaval: https://cdn.solarbotics.com/products/datasheets/mp2307.pdf. [Kasutatud 19 mai 2019].

[17] Infineon Techonlogies AS, „Andmeleht: P-Channel Mosfet,“ [Võrgumaterjal]. Saadaval: http://www.farnell.com/datasheets/1942417.pdf?_ga=2.13995646.878350690.1558309793-288096520.1537289289&_gac=1.95426670.1556116737.Cj0KCQjwkoDmBRCcARIsAG3xzl_uhMM5qippHkKPUAf1rl8jmrCM6kgVcap0uX9n0zd5AMjHdzxbtzEaAtnIEALw_wcB. [Kasutatud 19 Mai 2019].

[18] Premier Farnell, „Andmeleht: Bipolaartransistor BC635,“ [Võrgumaterjal]. Saadaval: http://www.farnell.com/datasheets/1693081.pdf?_ga=2.56200546.878350690.1558309793-288096520.1537289289&_gac=1.47634901.1556116737.Cj0KCQjwkoDmBRCcARIsAG3xzl_uhMM5qippHkKPUAf1rl8jmrCM6kgVcap0uX9n0zd5AMjHdzxbtzEaAtnIEALw_wcB. [Kasutatud 19 Mai 2019].

[19] Semiconductor Components Industries, „Andmeleht: Schottky dioodi SB540,“ [Võrgumaterjal]. Saadaval: http://www.farnell.com/datasheets/2299754.pdf?_ga=2.258172227.878350690.1558309793-288096520.1537289289&_gac=1.195425886.1556116737.Cj0KCQjwkoDmBRCcARIsAG3xzl_uhMM5qippHkKPUAf1rl8jmrCM6kgVcap0uX9n0zd5AMjHdzxbtzEaAtnIEALw_wcB. [Kasutatud 19 mai 2019].

[20] Littelfuse, Inc, „Andmeleht: Taastuv termokaitsme 30R250UU,“ 2010. [Võrgumaterjal]. Saadaval: http://www.farnell.com/datasheets/656941.pdf?_ga=2.13837822.878350690.1558309793-288096520.1537289289&_gac=1.22416073.1556116737.Cj0KCQjwkoDmBRCcARIsAG3xzl_uhMM5qippHkKPUAf1rl8jmrCM6kgVcap0uX9n0zd5AMjHdzxbtzEaAtnIEALw_wcB. [Kasutatud 19 mai 2019].

[21] „Tutvustus: STM32CubeMX,“ STMicroelectronics, [Võrgumaterjal]. Saadaval: https://www.st.com/en/development-tools/stm32cubemx.html. [Kasutatud 19 mai 2019].

[22] „Tutvustus: Atollic TrueStudio,“ [Võrgumaterjal]. Saadaval: https://atollic.com/truestudio/. [Kasutatud 19 mai 2019].

[23] „Tutvustus: FatFs,“ [Võrgumaterjal]. Saadaval: http://elm-chan.org/fsw/ff/00index_e.html. [Kasutatud 19 mai 2019].

[24] „Tutvustus: IDLE,“ [Võrgumaterjal]. Saadaval: https://docs.python.org/3/library/idle.html. [Kasutatud 19 mai 2019].

[25] Test Equipment Depot, „Manuaal: AFG-2005,“ [Võrgumaterjal]. Saadaval: http://www.testequipmentdepot.com/instek/pdf/afg-2000-series-manual.pdf. [Kasutatud 19 mai 2019].

LISAD

LISA 1 Pildid juhtplaadist ning kasutajaliidesest

LISA 2 Pilt võimendusplaadist

LISA 3 Katkestuste kood juhtplaadil

```
void HAL_TIM_IC_CaptureCallback(TIM_HandleTypeDef *htim)
{
  if(htim->Instance == TIM1)
  {
        if (htim->Channel == HAL_TIM_ACTIVE_CHANNEL_1)
        {
              IC2Value = HAL_TIM_ReadCapturedValue(&htim1, TIM_CHANNEL_1);
              if(IC2Value != 0)
              {
                    period1 = ((HAL_TIM_ReadCapturedValue(&htim1,
TIM_CHANNEL_2) * 100) / IC2Value );

                    pulse1 = 7200000 / IC2Value;
              }
        }
  }
  if(htim->Instance == TIM2)
    {
    if (htim->Channel == HAL_TIM_ACTIVE_CHANNEL_1)
    {
    IC2Value = HAL_TIM_ReadCapturedValue(&htim2, TIM_CHANNEL_1);
        if(IC2Value != 0)
        {
              period2 = ((HAL_TIM_ReadCapturedValue(&htim2, TIM_CHANNEL_2) *
100) / IC2Value );

              pulse2 = 7200000 / IC2Value;
        }
    }
    }
  if(htim->Instance == TIM12)
    {
    if (htim->Channel == HAL_TIM_ACTIVE_CHANNEL_1)
    {
    IC2Value = HAL_TIM_ReadCapturedValue(&htim12, TIM_CHANNEL_1);
        if(IC2Value != 0)
        {
              period3 = ((HAL_TIM_ReadCapturedValue(&htim12, TIM_CHANNEL_2) *
100) / IC2Value );

              pulse3 = 7200000 / IC2Value;
        }
    }
   }

  if(htim->Instance == TIM8)
    {
    if (htim->Channel == HAL_TIM_ACTIVE_CHANNEL_1)
    {
    IC2Value = HAL_TIM_ReadCapturedValue(&htim8, TIM_CHANNEL_1);
        if(IC2Value != 0)
        {
             period4 = ((HAL_TIM_ReadCapturedValue(&htim8, TIM_CHANNEL_2) *
100) / IC2Value );

              pulse4 = 7000000/ IC2Value;
        }
```

```
    }
   }


  if(htim->Instance == TIM5)
  {
        IC2Value = HAL_TIM_ReadCapturedValue(&htim5, TIM_CHANNEL_4);
        Kiirus1 = 7200000 / IC2Value;
        __HAL_TIM_SET_COUNTER(&htim5, 0);

  }

  if(htim->Instance == TIM14)
  {
        IC2Value = HAL_TIM_ReadCapturedValue(&htim14, TIM_CHANNEL_1);
        Kiirus2 = 7200000 / IC2Value;
        __HAL_TIM_SET_COUNTER(&htim14, 0);

  }
}
```

LISA 4 PWM signaali lugemise ja genereerimise kood

```
void Pwm_Generation(void)
{

	TIM_OC_InitTypeDef sConfigOC;

	HAL_TIM_IC_Stop_IT(&htim1, TIM_CHANNEL_2);
	HAL_TIM_IC_Stop_IT(&htim1, TIM_CHANNEL_1);
	HAL_TIM_IC_Stop_IT(&htim2, TIM_CHANNEL_2);
	HAL_TIM_IC_Stop_IT(&htim2, TIM_CHANNEL_1);
	HAL_TIM_IC_Stop_IT(&htim12, TIM_CHANNEL_2);
	HAL_TIM_IC_Stop_IT(&htim12, TIM_CHANNEL_1);
	HAL_TIM_IC_Stop_IT(&htim5, TIM_CHANNEL_4);
	HAL_TIM_IC_Stop_IT(&htim8, TIM_CHANNEL_1);
	HAL_TIM_IC_Stop_IT(&htim8, TIM_CHANNEL_2);

	htim12.Instance = TIM12;
	htim12.Init.Prescaler = 3599;
	htim12.Init.CounterMode = TIM_COUNTERMODE_UP;
	htim12.Init.Period = 100-1;
	htim12.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
	htim12.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE;
	if (HAL_TIM_IC_Init(&htim12) != HAL_OK)
	{
	  _Error_Handler(__FILE__, __LINE__);
	}
	htim2.Instance = TIM2;
	htim2.Init.Prescaler = 3599;
	htim2.Init.CounterMode = TIM_COUNTERMODE_UP;
	htim2.Init.Period = 100-1;
	htim2.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
	htim2.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE;
	if (HAL_TIM_IC_Init(&htim2) != HAL_OK)
	{
	  _Error_Handler(__FILE__, __LINE__);
	}
	htim1.Instance = TIM1;
	htim1.Init.Prescaler = 7199;
	htim1.Init.CounterMode = TIM_COUNTERMODE_UP;
	htim1.Init.Period = 100-1;
	htim1.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
	htim1.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE;
	if (HAL_TIM_IC_Init(&htim1) != HAL_OK)
	{
	  _Error_Handler(__FILE__, __LINE__);
	}
	htim8.Instance = TIM8;
	htim8.Init.Prescaler = 7199;
	htim8.Init.CounterMode = TIM_COUNTERMODE_UP;
	htim8.Init.Period = 100-1;
	htim8.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
	htim8.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE;
	if (HAL_TIM_IC_Init(&htim1) != HAL_OK)
	{
	  _Error_Handler(__FILE__, __LINE__);
	}

	sConfigOC.OCMode = TIM_OCMODE_PWM1;
```

```
        sConfigOC.Pulse = 0;
        sConfigOC.OCPolarity = TIM_OCPOLARITY_HIGH;
        sConfigOC.OCNPolarity = TIM_OCNPOLARITY_HIGH;
        sConfigOC.OCFastMode = TIM_OCFAST_DISABLE;
        sConfigOC.OCIdleState = TIM_OCIDLESTATE_RESET;
        sConfigOC.OCNIdleState = TIM_OCNIDLESTATE_RESET;
        if (HAL_TIM_PWM_ConfigChannel(&htim1, &sConfigOC, TIM_CHANNEL_1) !=
HAL_OK)
        {
          _Error_Handler(__FILE__, __LINE__);
        }
        if (HAL_TIM_PWM_ConfigChannel(&htim2, &sConfigOC, TIM_CHANNEL_1) !=
HAL_OK)
        {
          _Error_Handler(__FILE__, __LINE__);
        }
        if (HAL_TIM_PWM_ConfigChannel(&htim12, &sConfigOC, TIM_CHANNEL_1) !=
HAL_OK)
        {
          _Error_Handler(__FILE__, __LINE__);
        }
        HAL_TIM_PWM_Start(&htim1, TIM_CHANNEL_1);
        HAL_TIM_PWM_Start(&htim2, TIM_CHANNEL_1);
        HAL_TIM_PWM_Start(&htim12, TIM_CHANNEL_1);
        HAL_TIM_PWM_Start(&htim8, TIM_CHANNEL_1);
}
void Pwm_Read(void)
{
        MX_TIM1_Init();
        MX_TIM2_Init();
        MX_TIM12_Init();
        MX_TIM8_Init();
        HAL_TIM_IC_Start_IT(&htim1, TIM_CHANNEL_2);
        HAL_TIM_IC_Start_IT(&htim1, TIM_CHANNEL_1);
        HAL_TIM_IC_Start_IT(&htim2, TIM_CHANNEL_2);
        HAL_TIM_IC_Start_IT(&htim2, TIM_CHANNEL_1);
        HAL_TIM_IC_Start_IT(&htim12, TIM_CHANNEL_2);
        HAL_TIM_IC_Start_IT(&htim12, TIM_CHANNEL_1);
        HAL_TIM_IC_Start_IT(&htim8, TIM_CHANNEL_1);
        HAL_TIM_IC_Start_IT(&htim8, TIM_CHANNEL_2);
}
```

LISA 5 Signaali lugemise alamfunktsioon

```
void Signal_Logging(int Start, int Marker);
{
	i++;
	sprintf(FailiNimi_Kirjutamine, "Katse_%d_.csv", i);
	sprintf(FailiNimi_Kirj_Ekraanil, "Kirjutatakse faili : %s",
FailiNimi_Kirjutamine);
	hItem = WM_GetDialogItem(pMsg->hWin, ID_TEXT_12);
	TEXT_SetTextColor(hItem, GUI_GREEN);
	TEXT_SetFont(hItem, GUI_FONT_13B_1);
	TEXT_SetText(hItem, FailiNimi_Kirj_Ekraanil);

	res = f_mount(&SDFatFS, "", 1);
	if(res != FR_OK) {
	  Error_Handler();
	}

	f_open(&myFile, FailiNimi_Kirjutamine, FA_OPEN_ALWAYS|FA_WRITE|FA_READ);
	f_lseek(&myFile, f_size(&myFile));
	sprintf(buffwr5, "%s, %s \n",buffwr1, Kiirendus_C);
	f_printf(&myFile, "%s", buffwr5);
	Pwm_Read();
	Marker = 0;
	Start = 2;
    }
    if(Start == 2)
    {
	sprintf(buffwr2, "%d", period1);
	sprintf(buffwr3, "%d", period2);
	sprintf(buffwr4, "%d", period3);
	sprintf(buffwr6, "%d", period4);
	if(Marker)
	{
		f_printf(&myFile,"%s, %s, %s, %s, %s, %s, %s, %s \n", buffwr2,
buffwr3, buffwr4, buffwr6, Temp, Kiirus_C1, Kiirus_C2, Markerwr);
		Marker = 0;
	}
	else
	{
		f_printf(&myFile,"%s, %s, %s, %s, %s, %s, %s \n", buffwr2,
buffwr3, buffwr4, buffwr6, Temp, Kiirus_C1, Kiirus_C2);
	}
    }
    if(Start == 3)
    {
	f_close(&myFile);
	f_mount(0, "", 1);
	Pwm_stop();
    }
}
```

LISA 6 Signaali genereerimise alamfunktsioon

```
void Signal_Play()
{
	res = f_mount(&SDFatFS, "", 1);
	if(res != FR_OK) {
	  Error_Handler();
	}
	res = f_open(&myFile, FailiNimi_Lugemine, FA_READ);
	if(res != FR_OK){
		hItem = WM_GetDialogItem(pMsg->hWin, ID_TEXT_11);
		TEXT_SetTextColor(hItem, GUI_RED);
		TEXT_SetText(hItem, "Faili ei eksisteeri");

		hItem = WM_GetDialogItem(pMsg->hWin, ID_TEXT_12);
		WM_ShowWindow(hItem);
		hItem = WM_GetDialogItem(pMsg->hWin, ID_TEXT_10);
		WM_ShowWindow(hItem);
		hItem = WM_GetDialogItem(pMsg->hWin, ID_BUTTON_0);
		WM_ShowWindow(hItem);
		hItem = WM_GetDialogItem(pMsg->hWin, ID_BUTTON_2);
		WM_ShowWindow(hItem);
		hItem = WM_GetDialogItem(pMsg->hWin, ID_BUTTON_3);
		WM_ShowWindow(hItem);
		hItem = WM_GetDialogItem(pMsg->hWin, ID_BUTTON_4);
		WM_ShowWindow(hItem);
		hItem = WM_GetDialogItem(pMsg->hWin, ID_BUTTON_5);
		WM_ShowWindow(hItem);
		hItem = WM_GetDialogItem(pMsg->hWin, ID_BUTTON_6);
		WM_ShowWindow(hItem);
		hItem = WM_GetDialogItem(pMsg->hWin, ID_BUTTON_7);
		WM_ShowWindow(hItem);

		Loe = 3;
	}
	else
	{
		Pwm_Generation();
		f_gets(line, sizeof(line), &myFile);
		Loe = 2;
	}
	if(Loe == 2)
	{
		f_gets(line, sizeof(line), &myFile);
		if(Start == 3 || strlen(line) == 0 )
		{
			Loe = 0;
		  hItem = WM_GetDialogItem(pMsg->hWin, ID_TEXT_12);
		  WM_ShowWindow(hItem);
		  hItem = WM_GetDialogItem(pMsg->hWin, ID_TEXT_10);
		  WM_ShowWindow(hItem);
		  hItem = WM_GetDialogItem(pMsg->hWin, ID_BUTTON_0);
		  WM_ShowWindow(hItem);

		  hItem = WM_GetDialogItem(pMsg->hWin, ID_BUTTON_2);
		  WM_ShowWindow(hItem);
		  hItem = WM_GetDialogItem(pMsg->hWin, ID_BUTTON_3);
		  WM_ShowWindow(hItem);
		  hItem = WM_GetDialogItem(pMsg->hWin, ID_BUTTON_4);
```

```
                WM_ShowWindow(hItem);
                hItem = WM_GetDialogItem(pMsg->hWin, ID_BUTTON_5);
                WM_ShowWindow(hItem);
                hItem = WM_GetDialogItem(pMsg->hWin, ID_BUTTON_6);
                WM_ShowWindow(hItem);
                hItem = WM_GetDialogItem(pMsg->hWin, ID_BUTTON_7);
                WM_ShowWindow(hItem);
                period1 = 0;
                period2 = 0;
                period3 = 0;
                period4 = 0;
                Start = 0;
                f_close(&myFile);
                f_mount(0, "", 1);
                    break;
              }
        p = strtok(line, ",");
        period1 = atoi(p);
        htim1.Instance->CCR1 = period1;
        p = strtok(NULL, ",");
        period2 = atoi(p);
        htim2.Instance->CCR1 = period2;
        p = strtok(NULL, ",");
        period3 = atoi(p);
        htim12.Instance->CCR1 = period3;
        p = strtok(NULL, ",");
        period4 = atoi(p);
        htim14.Instance->CCR1 = period4;
}
```