EESTI INFOTEHNOLOOGIA KOLLEDŽ

Andres Käver

Mootorsõiduki parameetrite reaalajaline monitooring ja juhtimine

Diplomitöö

IT SÜSTEEMIDE ARENDUSE ÕPPEKAVA

Juhendaja: Raivo Sell, PhD

Konsultant: Kristiina Hakk, PhD

Toimetaja: Anu Kuusmaa, MA

Tallinn 2013

AUTORIDEKLARATSIOON

Deklareerin, et käesolev diplomitöö, mis on minu iseseisva töö tulemus, on esitatud Eesti Infotehnoloogia Kolledžile lõpudiplomi taotlemiseks IT süsteemide arenduse õppekaval. Diplomitöö alusel ei ole varem õppekava lõpudiplomit taotletud.

Autor: Andres Käver .....

(allkiri ja kuupäev)

Töö vastab kehtivatele nõuetele

Juhendaja: Raivo Sell .....

(allkiri ja kuupäev)

Sisukord

Lühendite ja mõistete loetelu

Antud töös on kasutatud mõisteid järgnevas tähenduses:

ADC (*Analog to Digital Converter*) – analoogsisendi teisendamine digitaalväärtuseks.
Aktuaator – elektromehhaaniline seade lineaarse liikumise saavutamiseks.
Arduino – levinud mikrokontrollerid ja sellega seotud arenduskeskkond.
ARM – soodsate ja väikese energiakuluga RISC-protsessorite sari.
ATMega – firma ATMEL 8-bitiste protsessorite perekond.
CAN (*Controller Area Network*) – kahejuhtmelise siini standard elektrooniliste seadmete ühendamiseks.
DAC (*Digital to Analog Converter*) – digitaalväärtuse teisendamine analoogväljundiks.
GPS (*Global Positioning System*)– globaalne positsioneermisseade.
IMU (*Inertial Measurement Unit*) – kombinatsioon erinevatest inertsanduritest.
RPM (*Rotations Per Minute*) – pöördeid minutis.
RTC (*Real Time Clock*) – iseseisva toitega reaalajaline kell.
SD-kaart – mälukaart andmete salvestamiseks.
Servo – miniatuurne täitur, tavaliselt kuni 180-kraadise ringliikumisega.
VRP – Verona Racing Parts, mootoritehas Itaalias.

Sissejuhatus

Käesoleva diplomitöö eesmärgiks on luua prototüüplahendus võistlusmootorsõiduki tegevusparameetrite analüüsiks ja juhtimiseks elektroonika, mehhatroonika ja infotehnoloogia vahendeid kasutades – ja saavutada seeläbi oluline resultatiivuse paranemine. Töö raames on plaanis välja töötada sobiv riistvaralahendus ja luua vastav tarkvara, mis oleks kasutajale kergesti kättesaadav, täiendatav ja kohandatav.

Teema valikul otsustas autor eelnimetatud eesmärgi kasuks, kuna saadaolevad kommertslahendused on kallid (maksavad tuhandeid eurosid), ligipääs terviklahendustele praktiliselt puudub, seadmete ning nendega seotud tarkvara modifitseerimine ei ole võimalik. Samuti on autor ise kaasatud tehnilise juhina mitmesse mootorispordi projekti ning huvitatud vastavatest rakendustest ning võimalikest edasiarendustest.

Aktuaalsus

Mootorsõidukite ja nende juhtide tegevusparameetrite objektiivne hindamine ja resultatiivsuse parandamine on olnud aktuaalne kogu mootorsõidukite ajaloo vältel. Enamikus on vastavad vahendid ja lahendused olnud tehniliselt keerukad või kohmakad kasutada ja kallid (tuhandetes eurodes) – seega enamasti väljaspool professionaalset ringkonda juurdepääsmatud. Mõõta näiteks kartauto 100 järjestikust starti ja teha järeldusi kiirenduse, lõppkiiruse, mootori

töötemperatuuride jmt parameetrite osas - kasutades stopperit ja termomeetrit - on teostatav, kuid ebamugav, ebatäpne ja aeganõudev.

Tänapäeval, kus kergelt programmeeritavad mikrokontrollerid on laialdaselt levimas, erinevaid analoog/digitaalsensoreid on olemas enamike vajalike suuruste mõõtmiseks, digitaalne andmeside on vabalt kättesaadav ja suured arvutusvõimsused ja andmemahud ei ole ka probleemiks laiatarbe personaalarvutitele, on võimalik teostada reaalajalise monitoorimise ja juhtimise ning telemeetria lahendusi võrdlemisi väikeste kuludega ja efektiivselt. Samuti on olemas kasutusvalmid teegid mikrokontrolleritele erinevate välisseadmete ja sensoritega suhtlemiseks, muutes vastavate seadmete arenduse lihtsamaks ja kiiremaks. Eelnevalt väljatoodud muutused loovad eelduse lahenduse teostamiseks odavate ja kättesaadavate vahenditega.

Diplomitöö sisu

Analüüsi osas annab autor konkreetse veemotospordi projekti näitel ülevaate, milliseid andmeid mootrsõidukis potentsiaalselt kogutakse, kuidas neid soovitakse analüüsida ja kasutada. Uuritakse, millised on mujal kasutatavad lahendused. Võrreldakse erinevate riistvara- ja tarkvarakomponentide sobivust soovitud resultaadi saavutamiseks. Analüüsi tulemusel autor leiab parimad lahendusviisid ning valib välja prototüübi realiseerimiseks sobiva riistvara- ja tarkvaralahenduse.

Teostuse osas kirjeldab autor diplomitöö praktilist poolt. Antakse ülevaade projekteeritud ja kasutatud riistvarast ning teostatud tarkvaralistest lahendustest.

Kokkuvõttes kirjeldatakse tehtud tööd ja kasutatud lahendusi. Lõpuks tuuakse välja diplomitööst saadav kasu ning edasine jätkuv uurimus- ja arendustöö.

Diplomitöö lisades on toodud skemaatilised joonised tehnilistest lahendustest ja programmikoodi näidised. Programmikoodi näidised on diplomitöö piiratud mahu

tõttu autori poolt valitud võimalikult illustratiivsed ja taaskasutatavad – kattes loodava prototüüpseadme baasfunktsionalsuse.

1. Analüüs

1.1 Tehniline lähteülesanne – veemotosport

O-125/250/350 klassi hüdroplaan tüüpi võistluspaate peetakse maailmas üheks konstruktsiooniliselt kõige lihtsamaks, kuid samas sooritusvõimelt kõige efektiivsemaks mootorispordi alaks. Eestis on O-klasside (O – *open*, ehk avatud/vabade reeglitega, kasutusel kahetaktilised mootorid kubatuuriga 125, 250 ja 350 cm^3) võistlussport pika ajalooga - veemotospordiga on Eestis tegeletud üle poole sajandi. 2011.a moodustati Andres Looritsa juhtimisel uus projektimeeskond eesmärgiga saavutada 2013.a maailmameistrivõistlustel tiitlivõit vähemalt ühes O-paadiklassidest. 2012.a suvel valmis ülimodernne O-250/350 komposiitpaadikorpus (joonis 1), 2013.a varasuvel valmib analoogne O-125 paadikorpus. Sooritusvõime optimeerimiseks ja maksimeerimiseks soovitakse kasutada modernseid mehhatroonika, elektroonika ja infotehnoloogia vahendeid. Mootorid, väljalasked ja jõuülekanded antud projektis ehitatakse koostöös VRP mootoritehasega Itaalias.

Joonis 1: O-250/350 klassi hüdroplaan, 2012.a suvi

Soovitud eesmärkide saavutamiseks on vajalik koguda ja analüüsida võimalikult paljusid mootorsõiduki dünaamilisi parameetreid.

Monitooritavad parameetrid:

- GPS – kiirus, suund, positsioon.
- IMU – kiirendused, kalded, kompass [32, 33, 34].
- Piloodipoolne paadi juhtimine – rool, gaas, väljalaske pikkus, mootorijala sügavus ja kalle.
- Muud tehnilised parameetrid – kütuse temperatuur, kütuse rõhk, mootori pöörded, väljalasete temperatuurid, iseseisev RTC-kellaaeg.

Kõik andmed tuleb jooksvalt lokaalselt salvestada SD-kaardile, tagamaks andmete terviklikkus hilisemaks analüüsiks.

Reaalajaline telemeetria – monitooritavad parameetrid tuleb edastada reaalajas kaldale ja kuvada sobival viisil. Maksimaalne distants 3km, enamikus otsenähtavus.

- GPS-info põhjal jooksev ringi- ja sektoriaegade võrdlus.
- Tagasiside sooritusvõime peenseadistamiseks.
- Statistiline andmete kogumine – kõik monitooritavad parameetrid salvestatakse andmebaasi võrdleva analüüsi teostamiseks.

Mootrsõiduki juhi informeerimiseks:

- Parameetrite visuaalne kuvamine näidikupaneelile.
- Alarmolukordade visualiseerimine.
- GPS-info põhjal automaatne ringiaja arvestus.

Sooritusvõime reaalajaline juhtimine monitooritavate parameetrite analüüsi abil:

- Kütusesegu juhtimine (elektrooniliselt juhitavad lisadüüsid karburaatoris).
- Automaatne mootori/sõukruvi kalde ja sügavuse juhtimine aktuaator-ajamite abil.
- Väljalaske pikkuse juhtimine servo abil.
- Gaasisiibrite juhtimine servo abil.

Antud teematika on tehnilistes detailides keerukas, suure mahuga ja ohutuskriitiline (saavutatavad kiirused kuni 200 km/h). Suurt tähelepanu tuleb pöörata kogutud mõõtmisandmete valiidsusele ja veaolukordade haldusele, samas tasakaalustades omavahel sooritusvõimet ja ohutust. Reaalne on oht nii kasutatava tehnika hävimiseks kui ka piloodi tervisele ja elule.

Autor valib edasise analüüsi käigus (vt peatükk 1.5 ja 1.6) diplomitöös käsitlemiseks eelnevalt toodud loetelust välja hinnanguliselt suurimat sooritusvõime kasvu pakkuvad punktid.

1.2 Analoogsed olemasolevad lahendused mujal maailmas

Võrreldavat funktsionaalsust pakkuvad enamkasutatavad tooted turul:

- Video VBOX Pro – GPSi salvestus 10 Hz, CanBus logimine, videokaamera salvestus. Puuduvad distantstelemeetria, sensorid ja väljundite juhtimine. Seadme komplekti hind alates 5000 eurost. [1]
- PerformanceBox – GPSi salvestus 10 Hz, CanBus logimine. Puuduvad distantstelemeetria, sensorid ja väljundite juhtimine. Seadme komplekti hind alates 1200 eurost. [2]
- Racepak G2Xpro – GPSi salvestus, CanBus logimine. Puudub distantstelemeetria, sensorid eraldi. Puudub väljundite juhtimine. Hind alates 1500 eurost. [3]
- AIM MXL05 – GPSi salvestus 5 Hz, CanBus logimine. Puudub distantstelemeetria, sensorid eraldi. Puudub väljundite juhtimine. Hind alates 2000 eurost. [4]

Pea kõiki turul saadaolevaid seadmeid iseloomustab distantstelemeetria võimaluse puudumine ja väljundite juhtimise puudumine – kaks peamist antud projektis püstitatud eesmärki. Samuti on võimatu või väga kallis personifitseeritud omaduste lisamine – seadmete elektroonikaskeemid ja seotud tarkvara lähtekoodid ei ole kättesaadavad.

Kogu projekti jaoks planeeritava lahendusega omadustelt ja sooritusvõimelt ligilähedane on Cosworth Inc poolt toodetav Pectel MQ12 seade baashinnaga 10 000 eurot. Antud tootega lähemalt tutvuda ei ole võimalik, seadet müüakse ainult kliendikohtumiste kaudu. [5]

1.3 Arendatavad lahendused

Inkrementaalse riistvaraarenduse käigus planeerib autor välja arendada mitu erineva komplekssusastmega ja universaalsusega andmekoguja lahendust eri keerukusastmega rakenduste tarbeks:

- Kompaktne, iseseisev monitooringumoodul, varustatud sisseehitatud näidikuga ja GPS-mooduliga. Välissensoritest ühendatavad kaks

temperatuurisensorit ja rpm-sensor. Moodul on ettenähtud kinnitamiseks sõiduki juhi vaatevälja. Tarkvaralahendus automaatsele ringiaja arvutusele. Andmete salvestus puudub (edaspidi LogerA).

- Eraldiseisev monitooringumoodul, piiramatu arvu sisenditega. Info jooksev lokaalne salvestamine mälukaardile. Distantstelemeetria. Eraldiseisev näidikutepaneel (edaspidi LogerB).
- Funktsionaalsus sama, mis eelnevas punktis. Lisandub reaalajaline väljundmoodul ja mehhatrooniline/elektrooniline osa konkreetse sõiduki parameetrite juhtimiseks (edaspidi LogerC).

Põhirõhk diplomitöös on suunatud LogerC kui parimat sooritusvõime tõstmist võimaldava lahenduse inkrementaalsele väljatöötamisele.

1.4 Arendatavad tarkvaralahendused

Eelnevas peatükis väljatoodud riistvaralahendused baseeruvad erineva jõudlusastme ja funktsionaalsusega mikrokontrolleritel ja nende programmeerimisel. Võimalusel üritab autor kasutada Atmel AVR protsessorite perekonda ja Arduino arenduskeskkonda – kui mikrokontrollerite hobiarendajatele õppimiseks ja edasiarendamiseks kõige väiksema raskusastmega ja kui turul üht kättesaadavamat. Jõudlusnõuete kasvades üle Arduino baasplatvormi poolt pakutava on planeeritud kasutusele võtta ARM-protsessorid ja nende arendusvahendid.

Andmete kuvamiseks, kogumiseks, visualiseerimiseks ja analüüsiks distantstelemeetriast ja/või andmekoguja mälukaardilt kirjutatakse autori poolt iseseisev tarkvaralahendus. Loodav lahendus peab võimaldama kaht suuremat funktsionaalsust – kiiret ülevaadet hetkeparameetritest ja varem kogutud testsessioonide andmete omavahelist võrdlust. Vastav tarkvaralahendus kuulub arendusele antud projekti hilisemates iteratsioonides, peale riistvaraliste prototüüplahenduste valmimist.

1.4 Süsteeminõuded

LogerA – funktsionaalsed nõuded:

- Hetkekiiruse, mootoripöörete ja kuni kahe eraldiseisva väljalaske temperatuuri kuvamine.
- Alarmid eelseadistatud piirväärtuste ületamisel.
- Automaatne ringiaja kalkuleerimine.
- Parima ringi aeg, automaatselt käivituv stopper.

LogerA – mittefunktsionaalsed nõuded:

- Pritsmekindel korpus ja nupud.
- Sisemine toiteallikas.
- GPS-andmehõive minimaalne kiirus 5Hz, soovitatavalt 20Hz.

LogerB – funktsionaalsed nõuded:

- Kõikide kogutavate parameetrite kuvamine.
- Alarmid eelseadistatud piirväärtuste ületamisel ohutuskriitiliste parameetrite puhul.
- Automaatne ringiaja kalkuleerimine.
- Parima ringi aeg, automaatselt käivituv stopper.
- Andmete salvestamine lokaalselt mälukaardile ja edastamine telemeetriale.

LogerB – mittefunktsionaalsed nõuded:

- Pritsmekindlad korpused ja nupud.
- Väline toiteallikas.
- GPS-andmehõive kiirus 20Hz.
- Häirekindel andmesideühendus logeri ja näidikuteploki vahel.

LogerC – funktsionaalsed nõuded:

- Kõikide kogutavate parameetrite kuvamine.
- Alarmid eelseadistatud piirväärtuste ületamisel ohutuskriitiliste parameetrite puhul.
- Automaatne ringiaja kalkuleerimine.

- Parima ringi aeg, automaatselt käivituv stopper.
- Andmete salvestamine lokaalselt mälukaardile ja edastamine telemeetriale.
- Mootorsõiduki kahetaktilise mootori reaalajaline juhtimine – väljalaske pikkuse muutmine, süüte ajastuse muutmine, karburaatori lisakütusedüüside magnetklappide kontrollimine.
- Veesõiduki muude parameetrite kontroll – jõuülekande (mootorijalg ja vint ehk sõukruvi) asend veesõiduki ja veepinna suhtes: ajami sügavus ja kalle.

LogerC – mittefunktsionaalsed nõuded:

- Pritsmekindlad korpused ja nupud.
- Väline toiteallikas.
- GPS-andmehõive kiirus 20Hz.
- Häirekindel andmesideühendus logeri, sensorite, telemeetria ja näidikuteploki vahel.

1.5 Esmased lahendatavad probleemid

Prototüüplahenduse inkrementaalseks väljatöötamiseks valib autor välja ja kirjeldab esmases iteratsioonis käsitlemist leidvad probleemid.

Suur probleem säde-süütega sisepõlemismootorite elektroonilisel juhtimisel on elektromagnetkiirgusest tulenevad häired, mis tugevalt segavad vastavate mikrokontrollerite tööd. Eriti häiritud on kõikvõimalikud andmesidelahendused perifeeriaseadmete ja erinevate protsessorite vahel. Samuti võib eeldada probleeme ADC mõõtmistulemustega ning ülinõrkpingega K-Type temperatuurisensorite kasutamisel väljalaskegaaside temperatuuri mõõtmisel. Elektroonikakomponentide valikul ja trükkplaatide koostamisel tuleb häirete vältimisele pidevat tähelepanu pöörata.

Suurimat jõudlusekasvu kahetaktiliste mootorite juures on saadud mootori väljalaskesüsteemi arendustöödega [6]. Suhteliselt lihtsa väljalaskesüsteemi optimeerimise teooria praktiline rakendamine kahetaktiliste mootorite juures on keerukas, baseerudes helilainete peegeldumisel. Suurimat tähtsust omab impulss-lainete tagasipeegeldamine, et need saabuksid silindri väljalaskeavani õigeaegselt

(lained levivad helikiirusel) ja tagaksid silindri korrektse täitumise põlemata küttеseguga. Baasvalem väljalaske teoreetilise pikkuse arvutamiseks avaldub kujul:

$$L = \frac{ED * 42545}{rpm}$$

kus
L – väljalaske ideaalne pikkus, millimeetrites (*length*),
ED – väljalaske tsükli pikkus kraadides, mõõdetuna väntvõlli pöördenurgana (*exhaust duration*),
rpm – mootori pöörded minutis, millele ideaalpikkuses väljalaset arvutatakse (pöörded maksimaalse jõu juures)(vaata [6]).

Konsultatsioon VRP tehase peainseneriga kinnitas, et väljalaske pikkuse muutmine vastavalt mootori hetkepööretele annab suurima teoreetilise võimaliku efektiivsuse kasvu. Väljalaske kuju ja pikkus on VRP poolt arvutatud ning testitud võimalikult efektiivselt toimivaks 12 000 kuni 13 000 pöörde juures minutis. Autori poolt 2012.a sügisel O-250 maailmameistrivõistlustel Slovakkias Sturovos sooritatud mõõtmised näitasid, et mootori pöörete töövahemik võistlussõidu käigus on keskmiselt vahemikus 9000-14000 rpm. Stardihetkel käivitatakse mootor ja sellega seotud ülekandesüsteem (sidur/käigukast on keelatud) paati osaliselt ahtri osas veest väljas hoides. Mootori pöörete maksimumi saavutamise momendil sõiduk vabastatakse (asetatakse vette). Sellel hetkel saavutatakse maksimaalne koormus ja mootori pöörded enne paadi korpuse glisseerimisfaasi saavutamist langevad hetkega 7000 pöörde juurde. Samuti langevad mootori pöörded kurvi läbimisel. Väljalaske pikkuse sobiv reguleerimine vastavalt hetkepööretele nendes olukordades tagaks mootori suurema efektiivsuse.

Sisepõlemismootori suurima efektiivsuse tagab üksikut kolvi töötsüklit vaadates optimaalse kütusesegu kasutamine. Kuna O-klassides kasutatakse võimsuse tõstmiseks kütusena spetsiaalseid metanooli ja õli segusid, siis lambda-andureid antud rakenduses kasutada ei saa (nagu kahetaktilistes mootorites üldiselt, tänu õli segamisele kütusesse). Mitte nii täpse, kuid üsna hea empiirilise indikatsiooni

mootori üldisest töörežiimi efektiivsusest annab väljalaskegaaside temperatuuri mõõtmine. Samas mootori töötemperatuuri tõustes üle ohutuskriitilise piiri (täpne suurus algselt teadmata) hävineb tavaliselt kogu põlemiskamber (kolb, silinder, küünal) (joonis 2).

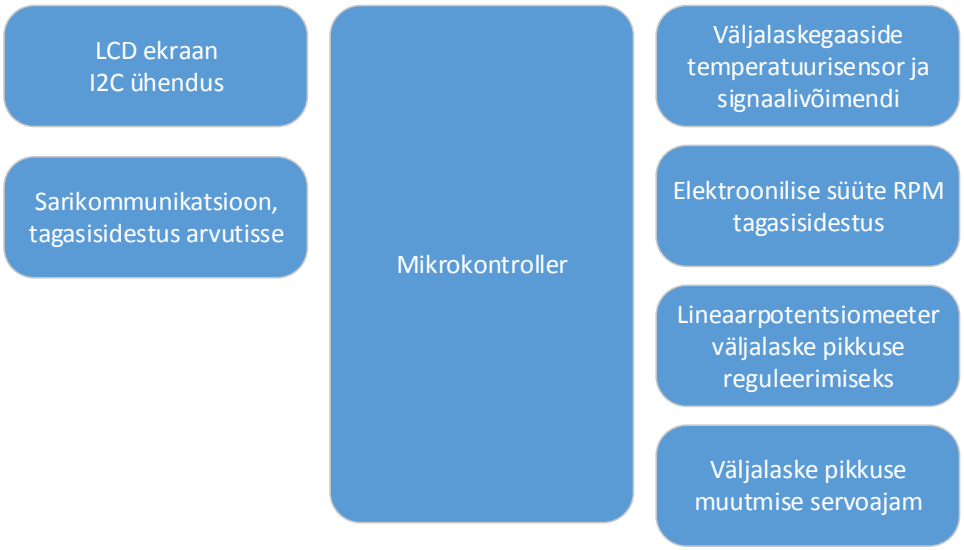
Joonis 2: Testpingis koormustestide käigus ohutuskriitilise temperatuuri ületanud mootori kolb

Toetudes eeltoodule, valib autor esmaseks lahendamiseks järgmised probleemid:

a) Mootori väljalaskegaaside temperatuuri mõõtmine.
b) Mootori pöörete (rpm) mõõtmine.
c) Optimaalse väljalaskepikkuse selgitamine kogu mootori pöörete vahemiku ulatuses.
d) Mootori efektiivseima töötemperatuuri ja ohutuskriitilise temperatuuri piiri tuvastamine.

Mootori vajalike optimaalsete tööparameetrite väljaselgitamise järel (seda käsitleb peatükk 2.9) saab järgmise realisatsioonisammuna välja töötada lahenduse (vt peatükk 2.10) mootori/väljalaske parameetrite juhtimiseks ja monitoorimiseks normaalolukorras.

1.6 Planeeritav esmane riistvaraarhitektuur



Joonis 3: Planeeritav esmane riistvaraarhitektuur

Joonisel 3 on esitatud planeeritav riistvaraarhitektuur mootori optimaalsete tööparameetrite mõõtmiseks testpingis. Sama arhitektuuriline lahendus (ilma liugtüüpi lineaarpotentsiomeetrita ja vastava tarkvaraga) võimaldab ka testida mootori tööparameetrite automaatset reaalajalist juhtimist parema sooritusvõime saavutamiseks.

2. Tehniline teostus

2.1 Mikrokontrolleri valik

Mikrokontrolleri (ennekõike küll selle protsessori) valiku aluseks on vajalike väljaviikude ja protokollide toetus (I2C, SPI, sariliidesed, digitaal- ja analoogsisendid, digitaalväljundid). Valiku üheks kriteeriumiks on ka platvormi kasutamise lihtsus edaspidises arenduses kolmandate osapoolte poolt.

Autori poolt testiti ja võrreldi nii AVR- kui ARM-mikrokontrollereid. Mikrokontrollerite loetelu on toodud tabelis 1.

Tabel 1
Mikrokontrollerite võrdlus

Nimetus/hind (€)	Protsessor	Tüüp	Arenduskeskkond
Arduino Uno R3 [7] 20	ATmega328p	16Mhz 8bit AVR	Arduino
Arduino Leonardo [8] 18	ATmega32u4	16Mhz 8bit AVR	Arduino
Arduino Mega [9] 39	ATmega2560	16Mhz 8bit AVR	Arduino
Arduino Due [10] 39	SAM3X8E	84Mhz 32bit ARM Cortex-M3	Arduino
Teensy 3.0[11] 25	Freescale MK20DX128	48Mhz 32bit ARM Cortex-M4	Arduino laiendus
chipKit Max32 [12] 50	Microchip PIC32MX795F512	80Mhz 32bit MIPS	Arduino laiendus
NXP LPC1769 [13] 24	LPC1769	120Mhz 32bit ARM Cortex-M3	Ainult C++

Vaadeldud mikrokontrollerite hinnad jäävad vahemikku 18 kuni 50 eurot. Kõik kontrollerid peale kõige lihtsamate Arduino Uno ja Arduino Leonardo pakuvad tuge suurele hulgale erinevatele sisenditele ja väljunditele. Kuna Arduino Mega, Arduino Due ja chipKit Max32 pakuvad kõige suuremat hulka sisendeid ja väljundeid ning on oma ühenduste paigutuselt kokkusobivad, siis valis autor esialgseks kontrolleriks Arduino Mega (alustades tööd LogerC arendamiseks). Eeliseks võrreldes Arduino Due'ga on 5-voldise pingega suure koormusetaluvusega sisendid/väljundid (kuni 40mA), samuti suur kogukonnapoolne tugi (IRC, foorumid, laialdane kasutamine hobi- ja kommertsprojektides). Suhteliselt hiljuti mikrokontrollerite turule tulnud ARM-tuumal baseeruvad protsessorid kasutavad 3,3V tehnoloogiat ja väga väikseid vooluvõimsusi (tüüpiliselt 3 kuni 9 mA) [10]. Väga väikesed võimsused ja madal nimipinge muudab protsessori sidumise sisend- ja väljundseadmetega tehniliselt oluliselt keerukamaks, nõudes vahekomponente ja lisatoitemooduleid. 8-bitise AVR-

platvormi protsessorite puuduseks on kindlasti madalam jõudlus, mis antud projekti lahendustes probleemiks muutudes nõuab mikrokontrolleri protsessoriplatvormi vahetust. Kasutades võimalusel väliseid spetsiaalseid sisend/väljundseadmeid (vältides kindla protsessoritüübiga seotud erilahendusi) ei tohiks platvormivahetus olla väga suur ja töömahukas probleem.

2.2 Temperatuurisensori signaalivõimendi valik

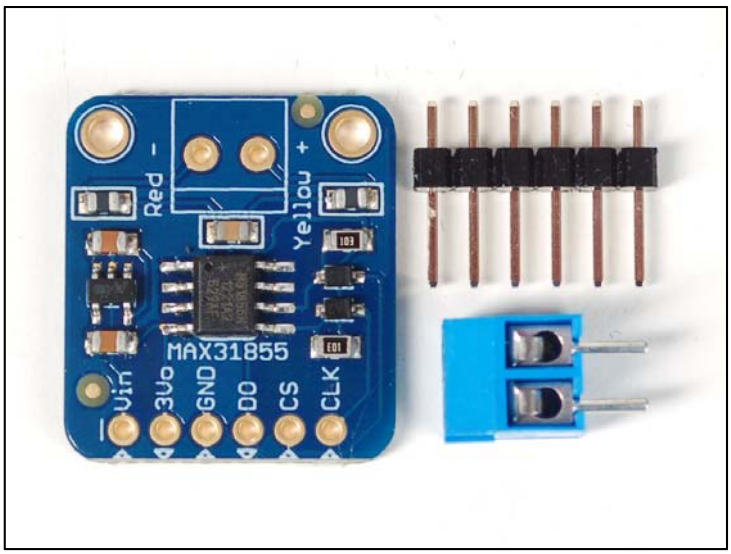
Sisepõlemismootorite töötemperatuur on üheks kõige ohutuskriitilisemaks parameetriks, seda eriti võistlusmootorite puhul. Konsultatsioonidel VRP tehase inseneridega juhiti ka autori tähelepanu suurtele probleemidele seoses säde-süütest tekkivate elektromagnetlainete ja K-Type temperatuurisensorite signaalivõimendi kasutamisega – väljalaskegaaside temperatuuri mõõtmine mootori suurtel pööretel on sageli andnud tulemuseks ebausaldusväärseid andmeid.

K-Type sensorid baseeruvad kahe erinevast metallist kokku keevitatud traadis tekkival pingel ja selle muutusel. Mõõdetav temperatuurivahemik on tüüpiliselt -200 kuni +1350°C. Probleemiks osutub väga väike pinge muutus: *ca* 50 µV temperatuuri ühekraadise muutuse kohta (1 µV on 1 / 1 000 000 volti), lisaks ei ole muutus kogu temperatuurivahemikus lineaarne. Kuna mikrokontrollerid ei suuda registreerida nii väikseid pingemuudatusi, siis kasutatakse taoliste sensorite puhul kas analoog- või digitaalsignaalivõimendeid (muundureid). Enamkasutatavad integreeritud kiibid K-Type temperatuurisensorite kasutamiseks on toodud tabelis 2.

Tabel 2
K-Type temperatuurisensorite signaalivõimendid

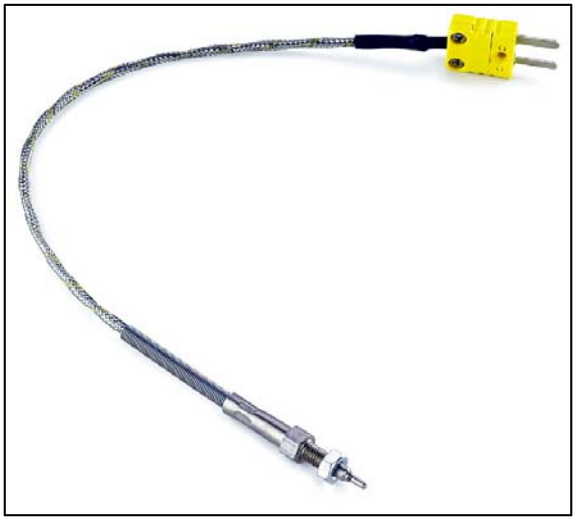
Tootja	Nimetus	Väljund	Täpsus (°C)	Info
Maxim Integrated [15]	MAX31855	Digitaalne	0,25	SPI-liides
Analog Devices [16]	AD595CQ	Analoog	1	10mV/°C
Analog Devices [17]	AD8495ARMZ	Analoog	3	5mV/°C

Esimeseks testimiseks valis autor MAX31855-põhise lahenduse (joonis 4). MAX31855-kiibil baseeruvat prototüüptestimiseks sobivat lahendust pakub Adafruit Industries [14].



Joonis 4: MAX31855 prototüüpplaat

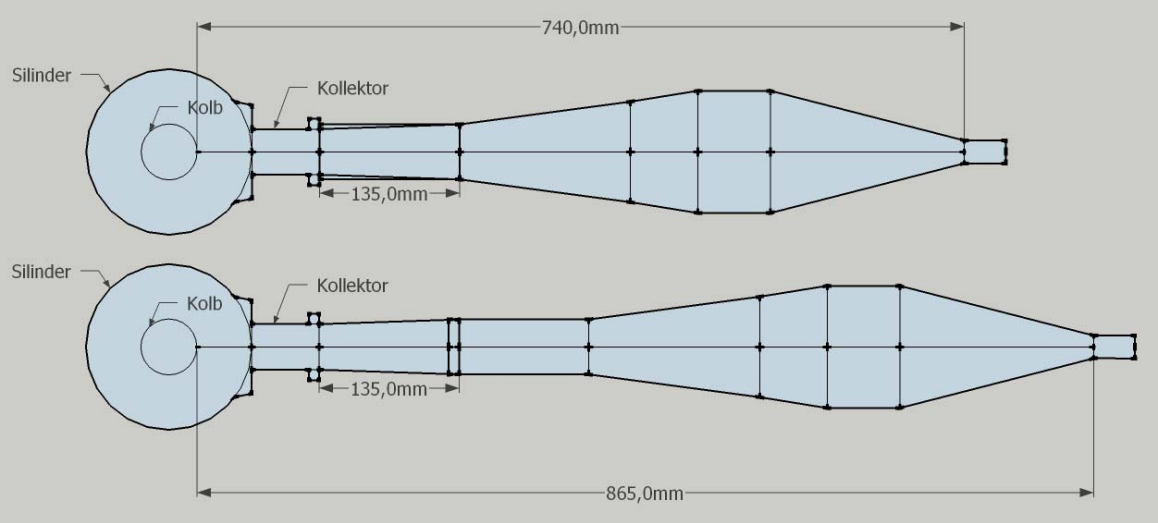
Mootori väljalaskegaaside temperatuuri mõõtmine seab vastavale K-Type sensorile kõrgendatud nõuded. Sensor peab olema vibratsioonikindel ja kaitstud häirete eest. Samuti peab sensoril olema sobiv kuju väljalaskekollektori külge kinnitamiseks. Autor valis kasutamiseks AIM Tecnologies (joonis 5) [18] poolt toodetava sensori. Sensor on spetsiaalse kujuga, kasutamiseks väljalaskekollektori külge keevitatud keermestatud läbiviigus (võimalikult lähedal silindri väljalaskeavale). Sensori tipp peab ulatuma 3-4 millimeetrit läbi kollektori seina.



Joonis 5: Väljalaskegaaside temperatuuri mõõtmiseks kasutatav K-Type sensor

2.3 Servoajam

Väljalaske pikkuse reguleerimise võimaldamiseks kasutatakse O-klassis kollektori teleskoopkonstruktsiooni vastavalt joonisele 6.



Joonis 6: Väljalaske pikkuse reguleerimine

Väljalaske pikkust on võimalik muuta vahemikus 740 mm kuni 865 mm – maksimaalne käigu pikkus on seega 125 mm (joonis 6). Mõõdetud maksimaalne heitgaaside poolt tekitatud lükkejõud 13 000 rpm juures on 50N.

Konsultatsioonidel EAMK [19] raadiomudelismi tehniliste spetsialistidega soovitati kasutada kollektori pikkuse reguleerimise ajamina mudelismis kasutuselolevaid suure võimsusega aktuaatoreid (servosid). Samuti on mikrokontrolleritel olemas riistvarapooleline tugi servode juhtsignaali genereerimiseks. Erinevate tootjate servomudelite parameetrite võrdlus on toodud tabelis 3. Põhjamaades kasutatakse valdavalt kahe firma toodangut: Futaba ja Hitec.

Tabel 3
Servode võrdlus

Tootja ja mudel	Kiirus/60Deg	Jõud (kg/cm)	Hind (€)
Futaba BLS157HV	0,11	21	130
Futaba BLS156HV	0,12	37	130
Hitec HS-7980TH	0,17	46	160
Hitec HS-7954SH	0,12	29	155

Parima kiiruse, jõu ja hinna suhte alusel otsutas autor kasutada Futaba BLS156HV servomootorit.

2.4 Mootori pöörete mõõtmine – riistvara

Konkreetses mootorispordi projektis kasutatakse firmas Zeeltronic eritellimusena valmistatud elektroonilist süütesüsteemi. Paadiklassis O-125 on kasutuses ühesilindriline mootor ja süütesüsteem Zeeltronic VRP-10. Paadiklassid O-250 ja O-350 kasutavad kahesilindrilist mootorit ja süütesüsteemi Zeeltronic VRP-20 [27].

Tüüpiliselt saab mootori pöördeid mõõta kahel viisil:

a) Süüteküünla ja süütepooli vaheliselt juhtmelt induktsiooni teel.
b) Elektrooniline süüde omab vastavat impulssväljundit.

Juhul, kui elektrooniline süüde omab vastavat väljundsignaali, on selle kasutamine alati eelistatum – signaalis on vähem häireid ja selle töötlemine tänu sellele lihtsam. Zeeltronic'i elektrooniline süüde omab vastavat väljundsignaali, signaali pinge on võrdne süüte toitepingega. Kuna kogu paadis olev elektroonika kasutab toiteks 16V LiPo (nelja paralleelse elemendiga) akuplokki, siis tuleb rpm-signaal enne mikrokontrollerit muundada sobivamaks. Kaaluda tuleb ka elektroonilisest süütest tulla võivaid häireid. Seoses sellega otsustas autor rpm-signaali optoisolaatoriga mikrokontrollerist eraldada. Kuna optoisolaator kasutab isolatsiooni tekitamiseks valgusdioodi, siis tuleb kasutada sobivat takistit voolu piiramiseks. Vastasel juhul hävineb optoisolaator koheselt. Autor valis kasutamiseks optoisolaatori FOD814 [28]. Vastavalt andmelehele on pingelang 1,2 volti ja tarbitav vool 20mA. Vastava vajaliku takisti väärtuse saab välja arvutada Ohmi seaduse abil [35].

$$R = \frac{Vs - Vl}{I}$$

kus

R – takistus,

Vs – sisendpinge,

Vl – pingelang,

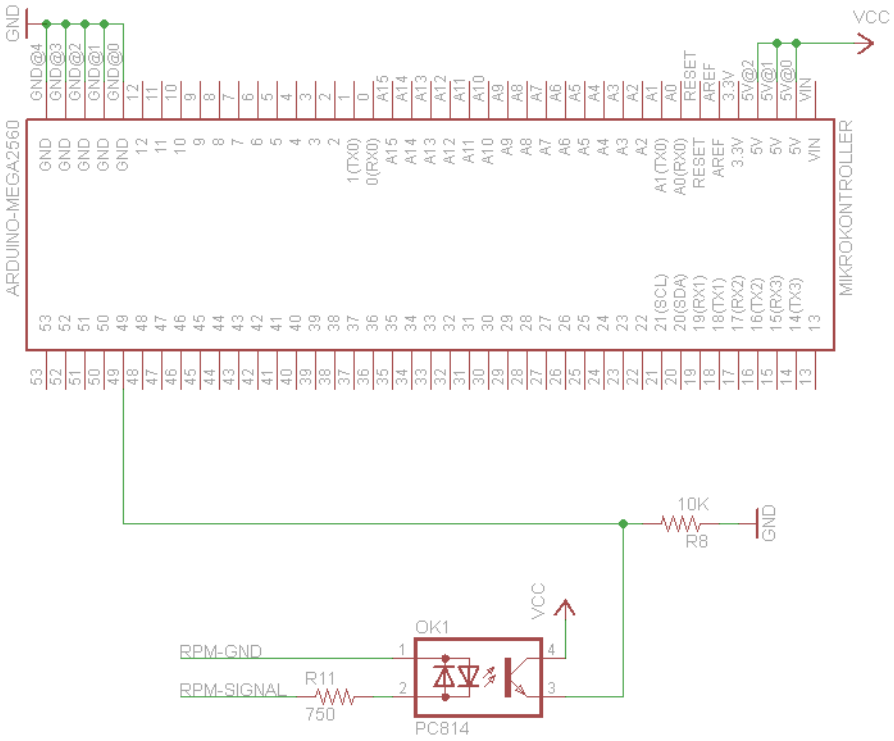
I – voolutugevus.

Konkreetse optoisolaatori puhul siis

$$R = \frac{16V - 1{,}2V}{0{,}02A} = 740\Omega$$

Takisti valikul tuleb kasutada lähimat suurema väärtusega standardtakistit. Antud puhul on selleks 750Ω.

Skeematiline lahendus on toodud joonisel 7.



Joonis 7: Mootori pöörete mõõtmine

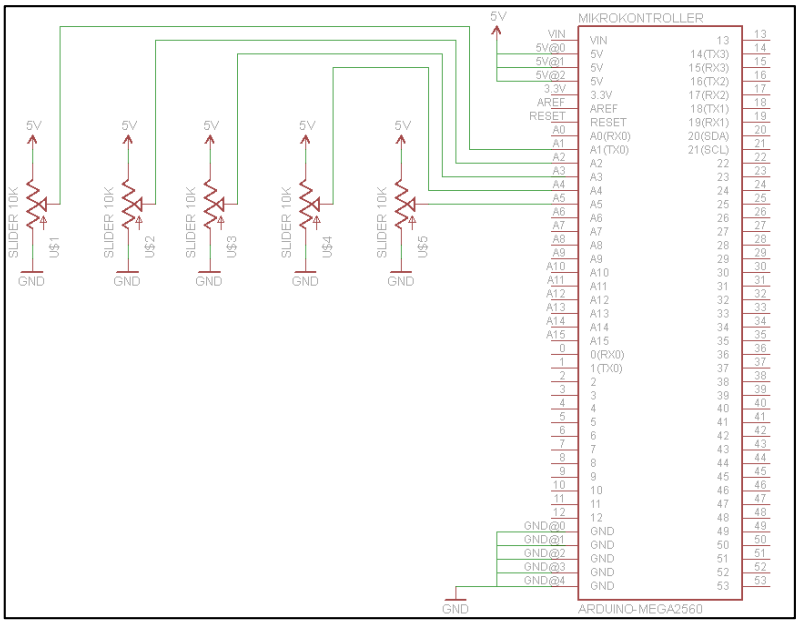
2.5 Muud riistvarakomponendid

Väljalaske pikkuse määramiseks sobib kõige paremini lineaarne liugpotentsiomeeter. Liugpotentsiomeetri kasutamine võimaldab potentsiomeetri juhthoova ja väljalaske pikkuse omavahelist lineaarset sidumist, muutes testimise käigus käsitsi kollektori pikkuse seadistamise intuitiivseks. Kuna potentsiomeetril baseeruvaid mõõtepunkte on mootorsõiduki juures mitmeid (gaasisiibri asend, rooli nurk jne), konstrueeriti autori poolt arendustööks vastavaid mõõtmisi mikrokontrollerile emuleeriv testseade (joonis 8).

Joonis 8: Potentsiomeetritega testrakis

Kasutatavate potentsiomeetrite mudel: PTA6043-2015DP-B103, tootjaks firma Bourns. Potentsiomeetrid on 60mm käiguga, 10kΩ nimitakistusega [22].

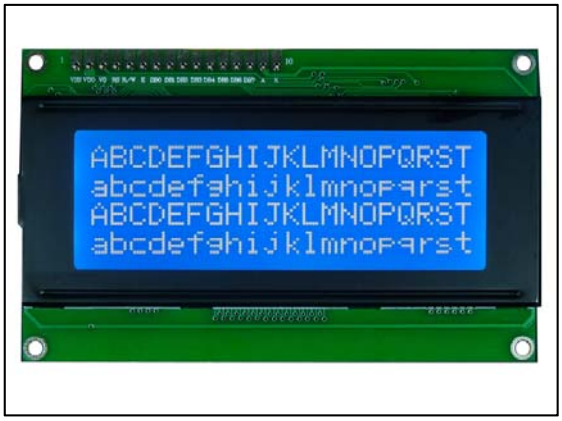
Potentsiomeetrite elektriline ühendusskeem mikrokontrolleriga on toodud joonisel 9.



Joonis 9: Potentsiomeetrite ühendusskeem

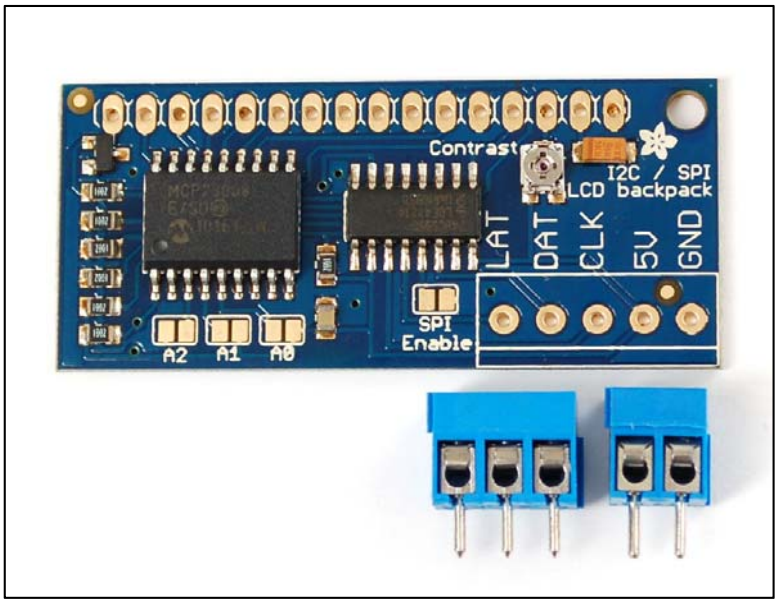
Kuvamaks süsteemi testimise käigus kogutavat infot võimalikult operatiivselt, otsustas autor kasutada mikrokontrollerite juures enamlevinud LCD-näidikut (näidik baseerub laialt toetatud Hitachi HD44780 juhtkiibil [25]). Valitud näidik suudab kuvada neli rida sümboleid, igas reas 20 tähemärki. Konkreetselt osutus

valituks mudel TC2004A-03 [23], kuvatud joonisel 10. Hitachi HD44780 juhtkiipi või selle analooge kasutavad LCD-näidikud on Arduino arenduskeskkonnas hästi toetatud, vastav LiquidCrystal teek on olnud arenduses alates 2008. aastast [24].



Joonis 10: LCD-näidik, 4x20 tähemärki

Kuna LCD-näidiku juhtimine mikrokontrolleri väljundite kaudu nõuab vähemalt kuue väljundi kasutamist, siis autor valis kasutamiseks ka I2C/SPI tuge pakkuva vahelüli, mis I2C-andmesideprotokolli kasutades vähendab hõivatud väljundite arvu kaheni. I2C-andmesideprotokoll/siin suudab korraga suhelda maksimaalselt kuni 112 välisseadmega. Vastav I2C/SPI-vahelüli on kuvatud joonisel 11.



Joonis 11: I2C/SPI LCD-näidiku vahelüli

Mooduli tootja ja vastava teegi arendaja on Adafruit Industries [26].

2.6 Esmase prototüüpseadme lõplik skeem

Mootori optimaalsete tegevusparameetrite mõõtmiseks koostatud prototüüpseadme lõplik skeem on toodud lisas 1.

2.7 Mootori pöörete mõõtmine

Mootori pöörete mõõtmiseks on võimalik kasutada mitut erinevat lahendust – eeldusel, et vastav impulss-signaal mootorist mikrokontrollerini jõuab. Mõned võimalikud meetodid rpm-signaali mõõtmiseks on: mikrokontrolleri väliste katkestuste teenindamise kasutamine (universaalne), mikrokontrolleri spetsiaalfunktsionaalsus (mikrokontrolleripõhine) või välise sagedus-volt-muunduri kasutamine. Autori poolt sooritatud testid näitasid, et mikrokontrolleri katkestuste või spetsiaalfunktsionaalsuse kasutamine segab mikrokontrolleri muud tööd, samuti on vajalik signaali elektrooniline eeltöötlemine häirete eemaldamiseks või vastava algoritmi väljatöötamine katkestuste teenindamisel. Spetsiaalselt mootori pöörete mõõtmiseks välja arendatud sagedus-volt-muundurid pakuvad juba sisseehitatud häire-eemaldus funktsionaalsust ja on oma töös väga stabiilsed. Kahetaktilise võistlusmootori juures võib osutuda probleemiks ADC-muunduri resolutsioon, mis on tüüpiliselt 10 bitti. See tähendab 1024 võimalikku väärtust. Võistlusmootor võib saavutada kuni 20 000 pööret minutis, mis muudab pöörete arvu mõõtmise ebatäpseks – eriti kui sagedus-volt-muundur on seadistatud ebakorrektselt (väljundpinge ei vasta mikrokontrolleri ADC pingevahemikule – tüüpiliselt 0 kuni 5 volti).

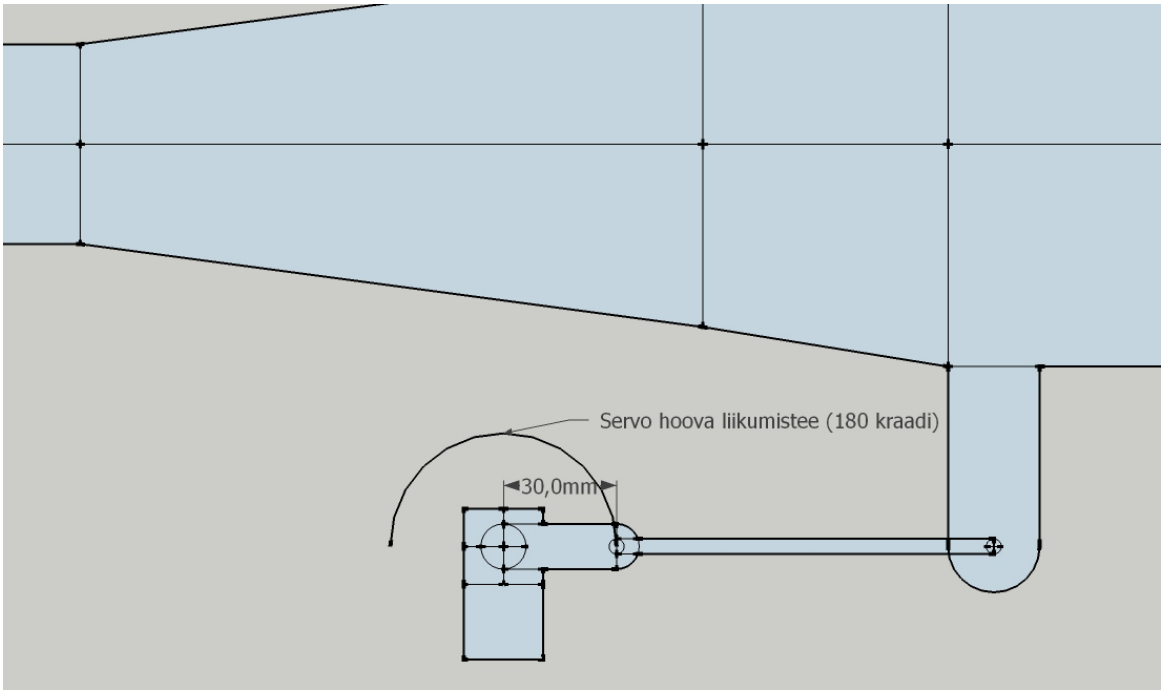
AVR Atmega protsessoritel baseeruvate mikrokontrollerite kasutamiseks on Paul Stoffregen kirjutanud populaarse teegi FreqMeasure, mis kasutab Atmega protsessorite eriomadusi sageduse mõõtmisel [29]. Juhul kui muid katkestusi nõudvaid meetodeid (servode juhtimine jmt) mitte pruukida või kasutada võimekamat protsessorit (Mega2560), saab antud teeki Atmega protsessorite puhul edukalt kasutada sagedus-volt-muunduri asemel. Prototüüplahenduse väljatöötamisel kasutas autor FreqMeasure teeki Arduino Mega2560 kontrolleril – lahendus töötab, kuid on mingitel tingimustel ebatäpne. Järgmises riistvara iteratsioonis on planeeritud võtta kasutusele riistvaraline sagedus-volt-muundur. Valida on mitme firma toodangu vahel, üks enamlevinumaid on Texas Instruments'i LM2907/LM2917-seeria integreeritud mikrokiip [30].

Kasutades FreqMeasure teeki, on vajalik implementeerida nii sisendsignaali tarkvaraline silumine kui ka olukord, kus sisendsignaal puudub (mootor seisab). Vastav pöörete kalkuleerimise funktsioon (Arduino 1.4, Arduino Mega2560 kontroller):

```
int rpmCalclulate(){
  // static - local variable, that retains its value inbetween calls
  static double rpmSum=0;
  static int rpmCount=0;
  static unsigned long lastRPMTime = 0;
  static int rpmFinal = 0;

  // input signal conditioning
  if (FreqMeasure.available()) {
    // time – initialized from millis() in every main loop cycle
    lastRPMTime = time;
    // average several readings together
    rpmSum = rpmSum + FreqMeasure.read();
    rpmCount = rpmCount + 1;
    if (rpmCount > 20) {
      // in rps/hz: (F_CPU / (sum / count))
      // multiply by 60, to get rpm
      //   Zeeltronic   VRP10   ignition   signal   has   double   peak,
      // so multiple by 30
      // F_CPU – clock rate of current board (16mhz)
      rpmFinal = (F_CPU / (rpmSum / rpmCount))*30;
      rpmSum = 0;
      rpmCount = 0;
    }
  }

  //if its timeout(>=1sek), then set rpm=0
  if ((rpmFinal>0) && (time-lastRPMTime >= 1000)) {
    rpmSum = 0;
    rpmCount = 0;
    rpmFinal = 0;
    lastRPMTime = time;
  }

   // return the current value of RPM
  return rpmFinal;
}
```

2.8 Mootori väljalaske pikkuse reguleerimine

Kahetaktilise mootori väljalaske pikkuse reguleerimine toimub väljalaske teleskooppikenemise teel (joonis 6). Servo hoova ringliikumine teisendatakse lineaarliikumiseks lihtsa mehhaanilise lahenduse teel vastavalt joonisele 12.



Joonis 12: Servo ja väljalaske tinglik mehhaaniline ühendus

Väljalaske joonpikkuse muutus tuleb arvutada funktsioonina servo hoova nurgast ja servo hoova pikkusest. Mootoritestid koormuspingis näitasid, et mootori pöörete töövahemikus (9000-13000 pööret minutis) on väljalaske pikkust vaja muuta maksimaalselt 60 mm ulatuses. Vajalik servohoova pikkus on pool sellest, seega kasutatakse 30 mm efektiivse pikkusega hooba. Arvutustes võib jätta arvestamata servohoova ja väljalaske vahelise vahelüli, tänu oma suurele pikkusele võrreldes servohoovaga. Punkti X ja Y koordinaate ringil arvutatakse järgmiste valemite abil:

$$X = X_0 + r * \cos(\alpha)$$

$$Y = Y_0 + r * \sin(\alpha)$$

kus

X_0 ja Y_0 on ringi keskpunkti koordinaadid,

r – ringi raadiuse pikkus,

α – raadiuse nurk vastupäeva liikudes x-teljelt.

Kuna väljalaske pikkus on määratud millimeetrites, servo juhtimisel aga kasutatakse servohoova nurka, siis tuleb antud X-koordinaadi arvutamise valemist avaldada nurk α:

$$\alpha = \arccos(\frac{X - X_0}{r})$$

Realisatsioonis peab arvestama aga füüsilisest maailmast tulenevate piirangutega. Neist esimene ja kõige olulisem on servo hoova piiratud liikumine, mis sageli on

väiksem kui maksimaalne 180 kraadi. Praktiliselt eeldab see kõigepealt servo hoova lõppnurkade kindlakstegemist (reaalne vahemik on sageli *ca* 10 kuni 170 kraadi) ja saadud väärtuste põhjal maksimaalse lineaarse liikumise väljaarvutamist.

$$X_{max} = r * cos(\alpha_{max})$$
$$X_{min} = r * cos(\alpha_{min})$$
$$X_{total} = abs(X_{max} - X_{min})$$

kus

r – servohoova pikkus,

α_{max} – servo maksimaalne saavutatav nurk,

α_{min} – servo minimaalne saavutatav nurk,

X_{max} – servohoova maksimaalne kaugus X-teljel 0-punkti suhtes,

X_{min} – servohoova minimaalne kaugus X-teljel 0-punkti suhtes (negatiivne väärtus)

X_{total} – servohoova lineaarne liikumisulatus.

Servohoova liikumisulatus on vajalik sisendparameetrite kontrollimiseks ja vigaste sisendite välistamiseks.

Vastav realisatsioon autori poolt kirjutatud Arduino teegi konstruktoris:

```
LinearServo::LinearServo(int servoArmMinAngle, int servoArmMaxAngle, int
servoArmLength){
  // set up the servo parameters
  _servoArmMinAngle = servoArmMinAngle;
  _servoArmMaxAngle = servoArmMaxAngle;
  _servoArmLength = servoArmLength;

  // calculate linear movement range acheivable wiht current servo configuration
  // angles are here in radians
  // this will be positive (right hand from centre)
  _servoMinX = _servoArmLength * cos(_servoArmMinAngle * degreesToRadians);
  // this will be negative (left hand from centre)
  _servoMaxX = _servoArmLength * cos(_servoArmMaxAngle * degreesToRadians);
  // maximum linear movement achievable
  _servoLinearRange = abs(_servoMaxX - _servoMinX);

}
```

Järgnevalt realiseeris autor eelpooltoodu põhjal soovitud lineaarse pikkuse alusel servohoova vajaliku nurga arvutuse funktsionaalsuse

```
int LinearServo::getServoAngleMath(int linearLength){
  // keep the required length in range
  if (linearLength > _servoLinearRange ){
```

```
    linearLength = _servoLinearRange;
  }
  if (linearLength < 0 ){
    linearLength = 0;
  }

  // move the required length into servo X coordinate system
  linearLength = linearLength + _servoMaxX;

  //calculate the required angle
  int res = acos(linearLength / (float) _servoArmLength) * radiansToDegrees;
  return res;
}
```

8-bitisel protsessoril (ATMega perekond) on ujukomatehted väga aeganõudvad – nii jagamine kui trigonomeetrilised tehted on ujukomatehted. Selle probleemi kontrollimiseks ja realiseerimiseks implementeeris autor sama funktsionaalsuse ka otsingutabeli põhjal.

Otsingutabeli täitmine töötsükli alguses:

```
// calculate and fill the exhaust servo length to angle conversion table
void LinearServo::servoAngleLookupTableFill(){
  for (int i=0; i <=_servoLinearRange; i++){
    _servoLookupTable[i] = getServoAngleMath(i);
  }
}
```

Funktsioon eelarvutatud vajalike servohoova nurkade tagastamiseks tabelist:

```
// lookup the exhaust servo angle
int LinearServo::getServoAngleLookup(int linearLength){
  // keep the required length in range
  if (linearLength > _servoLinearRange ){
    linearLength = _servoLinearRange;
  }
  if (linearLength <  0 ){
    linearLength = 0;
  }

  return _servoLookupTable[linearLength];
}
```

Kogu vastava teegi lähtekood asub lisas 2 ja lisas 3.

Sooritades Arduino Mega peal servoteegi kiiruse teste, sai autor järgnevad tulemused:

- Pikkuse igakordne arvutamine – 13,922064 sekundit.
- Pikkuse otsimine tabelist – 0,174548 sekundit.

Testimisel kasutati 60 000 väljakutset vastavatele funktsioonidele. Kiiruste vahe kahe meetodi puhul on 80-kordne. Vastavaid väärtusi arvutuslikult leides kulub

igakordselt tulemuse saamiseks 2,3 millisekundit – reaalajasüsteemide programmeerimise mõistes väga pikk aeg.

2.9 Optimaalsete mootori tööparameetrite mõõtmine

Kuna suurimat jõudlusekasvu kahetaktiliste mootorite juures on saadud mootori väljalaskesüsteemi arendustöödega [6], siis valis autor selle esmaseks lahendatavaks probleemiks. Kuna kahetaktilise mootori optimaalne väljalaske kuju on igal pöördel erinev, kasutatakse väljalaset, mis on arvutuslikult optimeeritud mootori kindlale pööretearvule. Konkreetne arv valitakse sõltuvalt mootorist, silindri kubatuurist, raja iseloomust jmt. Antud paadiklasside väljalase on optimeeritud töötama mootori 13 000 pöörde juures. 13 000 pööret on leitud mootoritootja poolt arvukate testide tulemusena olevat kõige optimaalsem. Testimaks väljalaske optimaalset pikkust kogu mootori töövahemiku pöörete ulatuses, realiseeris autor väljalaske pikkuse juhtimise liugpotentsiomeetri abil.

Vastav meetod potentsiomeetri väljundpinge lugemiseks mikrokontrolleri analoogsisendilt, kasutades aritmeetilist keskmist iga 10 lugemise järel

```
// read analog values, average over X readings
int exhaustOverrideCalculate(){
  static long sum=0;
  static int count=0;
  static int final = 0;

  sum = sum + analogRead(EXHAUST_OVERRIDE_PIN);
  count=count+1;
  if (count>=9){
    final = sum / count;
    sum = 0;
    count = 0;
  }

  return final;
}
```

Meetod tagastab väärtused vahemikus 0 kuni 1023 (Arduino ADC resolutsioon on 10 bitti), mis vastavad sisendpingele 0-5V.

Arduino programmi loop-meetodis teisendatakse potentsiomeetrilt saadud väärtus ümber väljalaske pikkuseks, mille põhjal omakorda arvutakse LinearServo teegi abil välja vajalik servohoova nurk. Vastav koodiosa:

```
//this is for engine bench testing, using sliding potentiometer
// for exhaust position
exhaustOverridePos = exhaustOverrideCalculate();
exhaustOverridePos = map(exhaustOverridePos, 4, 1021, 0, 1023);

// convert the exhaustOverridePos into millimeters
exhaustPos = map(exhaustOverridePos, 0, 1023, 0,
linearServo.getServoLinearRange());
// get the required servo angle
exhaustServoAngle = linearServo.getServoAngleLookup(exhaustPos);
exhaustServoAngle = map(exhaustServoAngle,servoMin,servoMax,0,180);

if (exhaustServoAngle!=exhaustServoAnglePrev){
  exhaustServoAnglePrev = exhaustServoAngle;
  exhaustServo.write((byte) exhaustServoAngle);
}
```

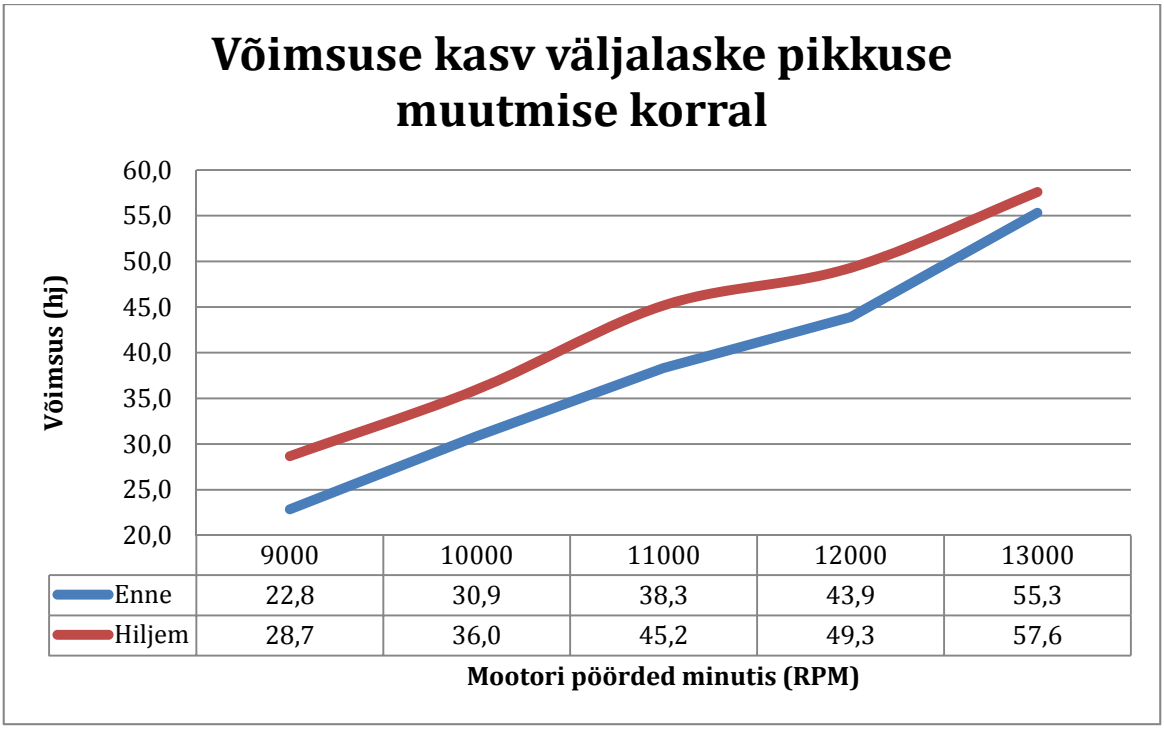
Kogu mootori optimaalse väljalaske pikkuse mõõdistamise programmikood on toodud lisas 4.

Tulemused mootori optimaalse väljalaske pikkuse mõõdistamisel testpingis mootori pöörete töövahemikus on toodud tabelis 4.

Tabel 4
Optimaalne väljalaske pikkus

Mootori pöörded	Väljalaske pikkuse muutus
8 000 ja alla	60 mm
9 000	60 mm
10 000	40 mm
11 000	25 mm
12 000	20 mm
13 000	5 mm
14 000 ja üle	0 mm

Saavutatud efektiivsuse kasv tänu väljalaske pikkuse dünaamilisele muutmisele on toodud joonisel 13.

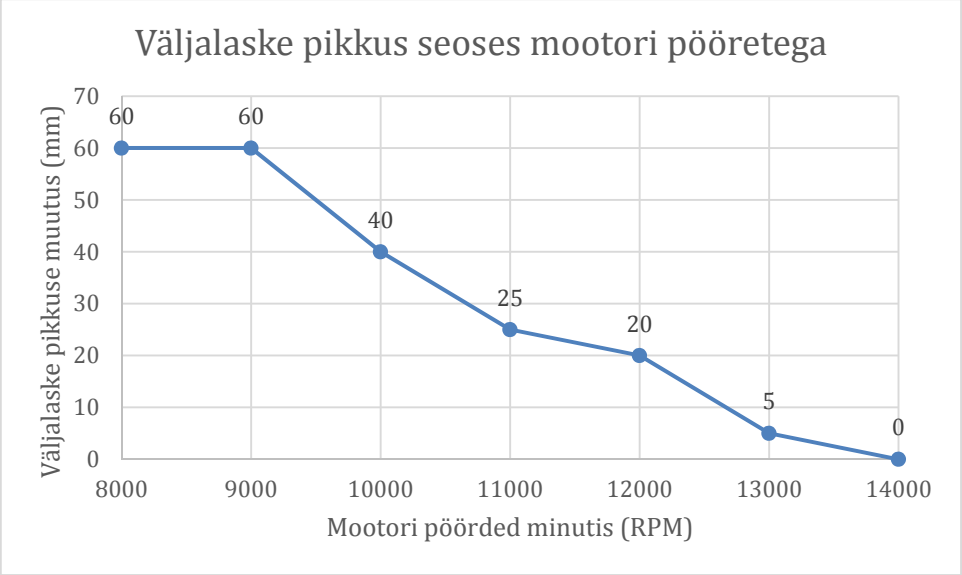


Joonis 13 : Mootori võimsuse kasv

Keskmiselt kasvas mootori võimsus 15 protsenti, madalate pöörete juures aga kasvas võimsus rohkem kui 25 protsendi. Võimsuse kasv sellises suurusjärgus annab selge konkurentsieelise ja eelduse püstitatud eesmärkide saavutamiseks.

2.10 Mootori efektiivsuse tõstmine tööparameetrite juhtimise abil

Suurimat reaalset efektiivsuse kasvu pakub väljalaske pikkuse dünaamiline reguleerimine vastavalt mootori hetkepöörete arvule. Ohutuskriitilise parameetrina peab siin jälgima väljalaskegaaside temperatuuri – efektiivsuse kasvades suureneb ka mootori töötemperatuur. Autori poolt sooritatud katsed VRP tehases mootorite testpingis näitasid optimaalseks kõrgeimaks töötemperatuuriks 340°C-360°C. 370-kraadise temperatuuri saavutamisel hakkas mootori temperatuur järsult kasvama ja mootori põlemiskamber/süüteküünal hävinesid ülekuumenemise tõttu (joonis 2). Joonisel 14 on graafiliselt kuvatud väljalaske pikkus seotuna mootori pööretega.



Joonis 14 : Väljalaske optimaalne pikkus seoses mootori pööretega

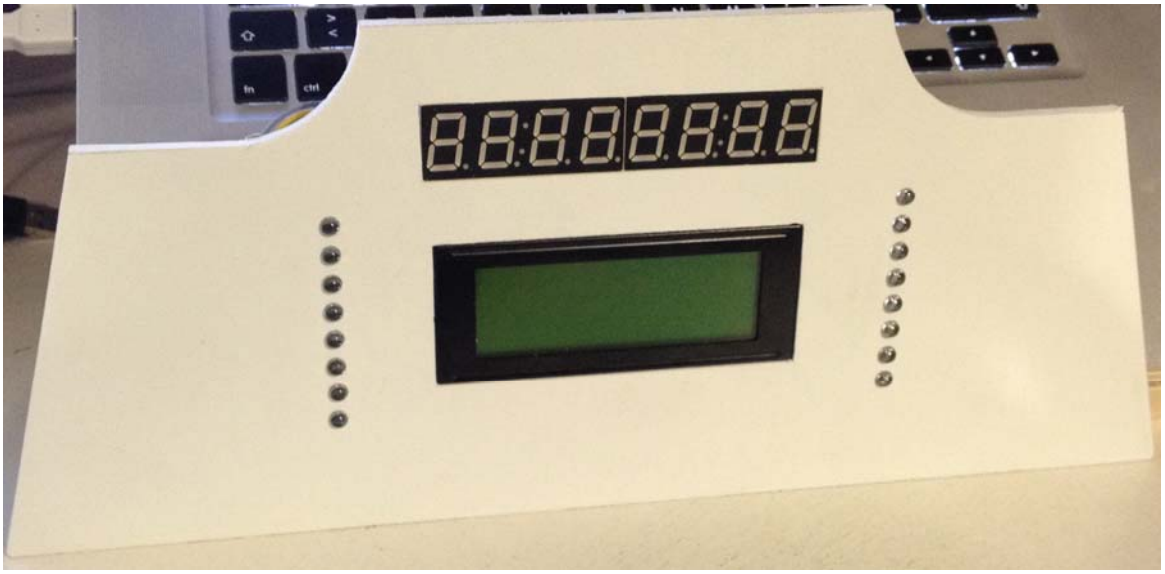
Joonise põhjal otsustas autor implementeerida lineaarse interpoleerimise vaheväärtuste arvutamiseks. Vastava funktsionaalsuse realisatsioon programmikoodis:

```
// mapping of rpm to optimum exhaust length
int engineRpmToExhLenMap[][2] = {
  { 8000,60},
  { 9000,60},
  {10000,40},
  {11000,25},
  {12000,20},
  {13000, 5},
  {14000, 0},
};

// calculate optimal exhaust length, based on current rpm
// and rpm to exhaust length mapping table
int CalculateExhaustPos(int currentRpm){
  // something wrong, return overall optimum length
  if (currentRpm<=0){
    return engineRpmToExhLenMap[engineRpmToExhLenMapSize-1][1];
  }

  // at or below minimum
  if (currentRpm<=engineRpmToExhLenMap[0][0]){
    return engineRpmToExhLenMap[0][1];
  }
  // at or over maximum
  if (currentRpm>=engineRpmToExhLenMap[engineRpmToExhLenMapSize-1][0]){
    return engineRpmToExhLenMap[engineRpmToExhLenMapSize-1][1];
  }

  // go over the mapping table and find suitable range
  int i=1;
  for ( ; i<engineRpmToExhLenMapSize; i++){
    if (engineRpmToExhLenMap[i][0]>=currentRpm){
      break;
    }
  }
```

```
  int res = map(currentRpm,
    engineRpmToExhLenMap[i-1][0],engineRpmToExhLenMap[i][0],
    engineRpmToExhLenMap[i-1][1],engineRpmToExhLenMap[i][1]);

  return res;
}
```

Võimalikus mootori pöörete lugemisel tekkivas veaolukorras viiakse väljalase lühimasse asendisse, tagades nii suurima keskmise sooritusvõime.

Samuti võttis autor kasutusele MAX31855 (täpsemalt vaata peatükist 2.2) integreeritud kiibil baseeruva temperatuurimõõtmise lahenduse väljalaskegaaside temperatuuri monitoorimiseks.
Läbi mitme iteratsiooni valmis ka klassi O-250 näidikutepaneel, kuvatud joonisel 15.



Joonis 15 : Paadiklassi O-250 näidikutepaneel

Esialgne planeeritud infopaigutus kesksel LCD-paneelil on toodud joonisel 16.

Joonis 16 : Infoväljade paigutus LCD-paneelil

Kuvatakse mootori pöördeid, väljalaske pikkust, väljalaske hetke- ja maksimumtemperatuuri (paadiklassi O-250 mootor on kahesilindriline, vajalikud on eraldi näidud LT ja RT), süütekaardi valikut (tulevane arendus), karburaatori lisadüüside kasutamist (tulevane arendus), gaasisiibri asendit ja süsteemi tööaega sekundites.

2.11 Lahenduse edasine optimeerimine

Testid diplomitöös kirjeldatud prototüüpseadmega näitasid mitut probleemi süsteemi sooritusvõime ja tehniliste omadustega. Üks suuremaid probleeme on tehted ujukomaarvudega, mis 8-bitistel ATMega protsessoritel on väga aeganõudvad. Programmi järgmises iteratsioonis planeerib autor üle minna täisarvudel põhinevale teostusele ja kasutada võimalikult palju ettearvutatud väärtusi.

Väga suured jõudlusprobleemid on MAX31855 kasutamisel, kus iga temperatuuriinfo lugemine võtab aega 200 millisekundit. Alternatiivse lahendusena planeerib autor kasutusele võtta AD595AQ [31] integreeritud kiibil baseeruva lahenduse. AD595AQ omab pidevat analoogväljundit, mille lugemisvõimekus on mikrokontrolleril endal väga hea.

Probleemkohaks on ka servo ebatäpne juhtimine ühekraadiste sammude kaupa (0 kuni 180). Servo reaalne juhtsignaal on pulsilaiusmodulatsiooniga, pakkudes kuni 10-bitist eraldusvõimet. Järgmises iteratsioonis on planeeritud kasutusele võtta otsene servo juhtsignaali kasutamise funktsionaalsus.

3. Edasine tegevus

Järgnevate riist- ja tarkvaraarenduse iteratsioonidega valmib LogerC (peatükk 1.3) esmane, enamikku planeerituid funktsionaalsusi sisaldav prototüüp. Lisandub GPS-jälgimine ja distantstelemeetria. Planeeritav valmimisaeg ja testimine on mai lõpus ja juuni alguses 2013. Valmimas on ka vastav tarkvara tugimeeskonnale strateegiliste otsuste langetamiseks. Esimene reaalne võrdlus ja saavutatud resultatiivuse hindamine toimub juunis 2013 Poolas maailmameistrivõistlustel paadiklassile O-125. Paadiklassile O-250/350 on valmimas ka kogu jõuajami sügavuse ja kalde reguleerimise mehhanism ja selle juhtimine. See süsteem jõuab testimisse 2013. aasta juulis ja esimestele võistlustele 2013. aasta juuli lõpus, O-250 klassi maailmameistrivõistlustel Itaalias.

Kokkuvõte

Käesoleva diplomitöö eesmärgiks oli luua prototüüplahendus võistlusmootorsõiduki (ja selle juhi) tegevusparameetrite analüüsiks ja juhtimiseks elektroonika, mehhatroonika ja infotehnoloogia vahendeid kasutades – saavutades seeläbi sooritusvõime paranemise.

Analüüsi osas autor võrdles muid taolisi lahendusi maailmas, võrreldes nende tehnilisi parameetreid ja omadusi, puudujääke ja hindu. Püstitati esialgsed nõuded loodavale süsteemile, esitades kolm erineva funktsionaalsusvõimekusega riistvarakomplekti lahendust. Tutvustati diplomitöö aluseks olevat mootorispordiklassi – Open-tüüpi hüdroplaane ja nendega seotud eesmärke – vähemalt ühe maailmameistritiitli võitmist 2013. aastal.

Praktilises osas, toetudes varasemale analüüsile, autor võrdles ja valis töö käigus välja vajalikud esmased riistvarakomponendid, koostas vajalikud elektroonikaskeemid ja ehitas prototüüplahenduse. Valminud prototüüplahendus hõlmab mootori pöörete ja väljalaskegaaside temperatuuri mõõtmise, väljalaske pikkuse muutmise vastavalt mootori pööretele ja näidikutepaneeli lahenduse.

Samuti arendati välja sobiv tarkvaralahendus kahetaktilise võistlusmootori koormustestimiseks ja andmete kogumiseks. VRP mootoritehases Itaalias Veronas nelja päeva jooksul teostatud testide ja sealt kogutud andmete põhjal koostati riist- ja tarkvaralahendused mootori juhtimiseks reaalkasutuses.

Prototüüplahenduse väljatöötamise käigus ilmnesid mitmed edasist optimeerimist võimaldavad kohad, mistõttu peab autor vajalikuks esmalt vaadata üle erinevate algoritmide teostus (eemaldada ujukomatehted) ja nende sobivus kasutatava mikrokontrolleriga. Samuti peab autor vajalikuks kogu prototüüplahenduse töötäpsuse tõstmist ja põhjalikumat ohutuskriitilist testimist.

Olulise mõõdetava tulemusena saavutas autor koostatud prototüüpseadme abil väljalaske pikkuse juhtimise teel mootori võimsuse kasvu keskmiselt 15%, madalamatel pööretel aga rohkem kui 25%. Samuti mõõdistas autor mootori sobiva töötemperatuuri ja selle ohutuskriitilise piiri – võimaldades edaspidi täpsemat mootori seadistamist ja selle töö monitoorimist ning ohutuse tagamist.

Diplomitöö käigus realiseeritud lahendused on autori hinnangul heaks baasiks muudele taolistele projektidele, kus on vaja seadmete dünaamilisi tööparameetreid koguda, analüüsida ja neile reageerida. Samuti saavutas autor olulise rahalise kokkuhoiu, hinnanguliselt on kõige keerukama seadme hind umbes 20 korda odavam kui samalaadne kommertslahendus – omades samas rohkem lisavõimalusi (Cosworth Pectel MQ12 hind algab 10 000 eurost, valminud prototüüplahenduse hind ei ulatu ka edasiste arenduste käigus üle 500 euro).

Eesti Veemotospordi Liit on tellinud seadme lihtsama versiooni kõigile oma noorteklassi sõidukitele.

Summary

Real-time Monitoring and Guidance of Motorized Vehicle Parameters. Diploma thesis by Andres Käver.

Purpose of this diploma thesis was to create a prototype solution for monitoring, analyzing and guiding motorized competition vehicle and its driver's performance, by the means of electronics, mechatronics and IT solutions – therefore achieving performance improvements.

In the analysis section author compared similar solutions available in the market, comparing their technical parameters, features, shortcomings and prices. Preliminary requirements where presented, describing three hardware and software sets with different functional capabilities. Author introduced Open type hydroplane competition boat, on which the thesis theory and implementation is based. Main aim is to win at least one world championship title in Open class during the year 2013.

In the implementation section, based on previous analysis, author compared and made choices about various hardware components, made required schematics for electronics and built up first hardware and software prototype. Prototype has following functionality: RPM sensing, measuring exhaust gas temperature, dynamic exhaust length regulation and dashboard display solution.

Also software solution for load testing and test-data gathering of two-stroke engines was developed. After four days of testing, data gathering and data analysis in engine factory (VRP, Italy) author designed hardware and software for engine guidance in real-world competition usage.

During prototyping phase author discovered several optimization opportunities. Author's opinion is that most important is redesign of current algorithms (removal of floating point calculations) and evaluating their suitability with currently used microcontroller. Also, much attention has to be paid to the speed, reliability and safety critical aspects of the whole system.

Important measurable metric achieved with the help of prototype solution was average increase of 15% of the engine power output and in the lower RPMs power the increase was more than 25%. Author also measured optimum working and critical safety temperatures of the engine – enabling more exact optimization and monitoring in the future.

Solutions developed in the thesis are in author's opinion good base for other similar projects - where measuring, data collection, data analysis and guidance of the motorized equipment is needed. Also substantial monetary savings – up to 20 times - were achieved. Total price of the solution, even with the future developments, should not exceed 500 euros (Coswoth Pectel MQ12 price starts at 10 000 euros).
Currently Estonian Water Motorsports Club has ordered the basic version of the prototype solution to all its youth class boats.

Kasutatud kirjandus

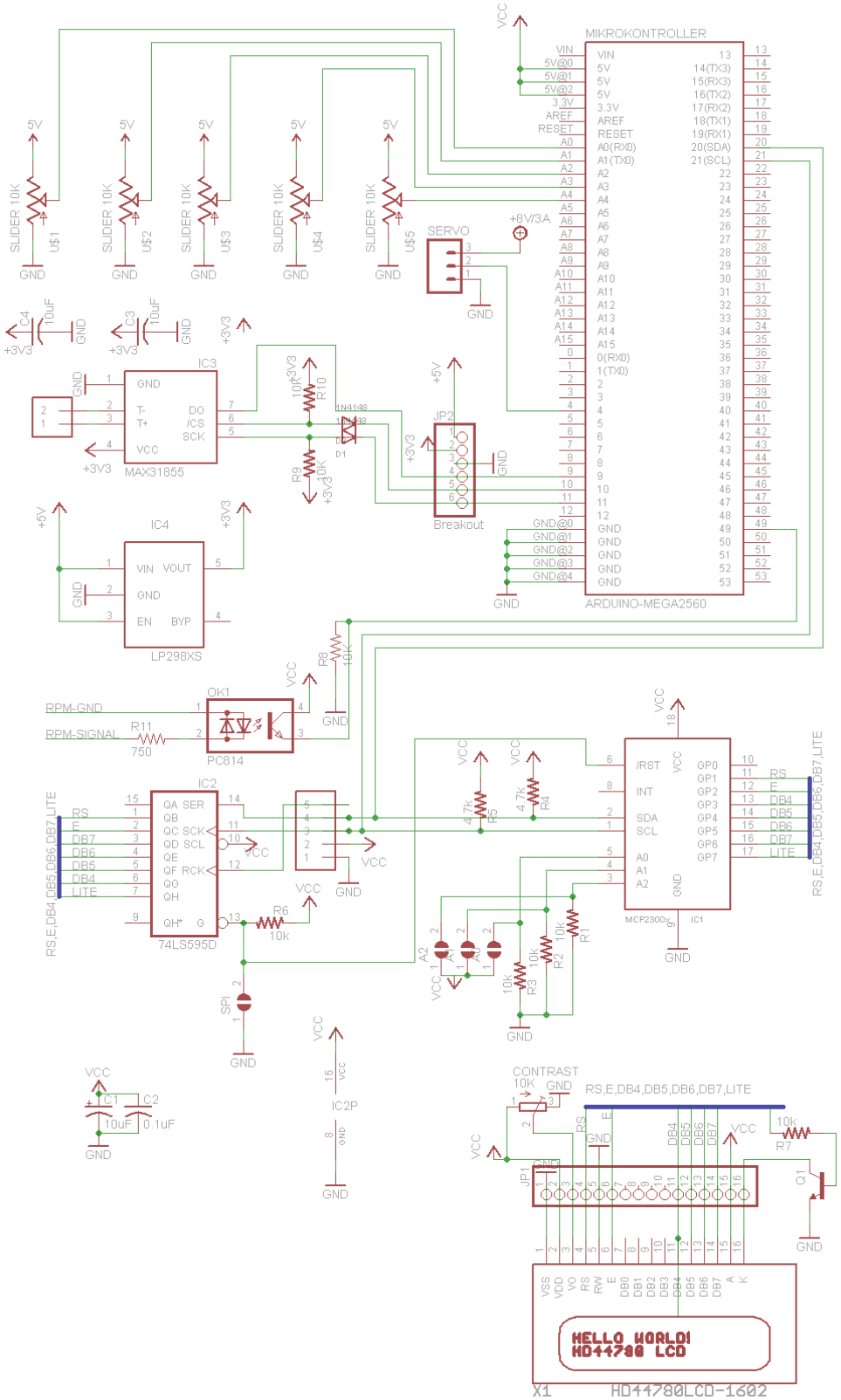
1. Racelogic (27.02.2013) VBOX pro [http://www.racelogic.co.uk/Store/categories.php?category=Video-VBOX/Video-VBOX-Pro/Video-VBOX-Pro-Packages]
2. Racelogic (27.02.2013) PerformanceBox [http://www.racelogic.co.uk/Store/categories.php?category=Performance-Meters/Performance-Box-Sport/PerformanceBox-Sport-Packages]
3. Racepak (28.02.2013) G2X [http://www.racepak.com/Loggers/G2X_Pro.php] [http://www.racepak.com/Price_List.pdf]
4. Aim Technologies (28.02.2013) MLX Pro 05 [http://shop.aim-tec.co.uk/aim-mxl-pro-05-dash-data-logger-for-data-acqusition-12-p.asp]
5. Cosworth (28.02.2013) Pectel MQ12 [http://row.cosworth.com/products/marine/powerboat-racing/pectel-mq12/]
6. A.Graham Bell (1998.a) Two-Stroke Performance Tuning,. 2. Edition.
7. Arduino (21.04.2013) Uno R3 [http://arduino.cc/en/Main/ArduinoBoardUno]
8. Arduino (21.04.2013) Leonardo [http://arduino.cc/en/Main/ArduinoBoardLeonardo]
9. Arduino (21.04.2013) Mega [http://arduino.cc/en/Main/ArduinoBoardMega2560]

10. Arduino (21.04.2013) Due
[http://arduino.cc/en/Main/ArduinoBoardDue]
11. Pjrc (21.04.2013) Teensy 3.0 (21.04.2013)
[http://www.pjrc.com/teensy/]
12. Digilent (21.04.2013) chipKit Max32
[http://www.digilentinc.com/Products/Detail.cfm?Prod=CHIPKIT-MAX32]
13. LPC (21.04.2013) NXP LPC1769
[http://www.lpctools.com/lpc1768.lpcxpresso.aspx]
14. Adafruit (01.03.2013) Thermocouple Amplifier MAX31855 breakout board
[http://www.adafruit.com/products/269]
15. Maxim Integrated (01.03.2013) MAX31855
[http://datasheets.maximintegrated.com/en/ds/MAX31855.pdf]
16. Analog Devices (01.03.2013) AD595
[http://www.analog.com/static/imported-files/data_sheets/AD594_595.pdf]
17. Analog Devices (01.03.2013) AD8495
[http://www.analog.com/static/imported-files/data_sheets/AD8494_8495_8496_8497.pdf]
18. AIM Technologies (22.04.2013) K-Type EGT Sensor
[http://shop.aim-tec.co.uk/egt-sensor-387-p.asp]
19. Eesti Automudelispordi Klubi (22.04.2013)
[http://www.modelcar.ee/3.php]
20. Futaba (22.04.2013) Futaba Brushless Servos
[http://www.futaba-rc.com/servos/brushless.html]
21. Hitec RCD (22.04.2013) Hitec Servos
[http://www.hitecrcd.com/products/servos/digital/hv-ultra-premium-digital/index.html]
22. Farnell (23.04.2013) 10K liugpotentsiomeeter
[http://www.farnell.com/datasheets/91892.pdf]
23. Adafruit (23.04.2013) LCD Näidik, TC2004A-03
[http://www.adafruit.com/datasheets/TC2004A-01.pdf]
24. Adafruit (23.04.2013) LiquidCrystal teek
[http://arduino.cc/en/Tutorial/LiquidCrystal]
25. Hitachi (23.04.2013) HD4478 andmeleht

[http://lcd-linux.sourceforge.net/pdfdocs/hd44780.pdf]
26. Adafruit (23.04.2013) i2c/SPI character LCD backpack [http://www.adafruit.com/products/292]
27. Zeeltronic (24.04.2013) elektrooniline süüde [http://www.zeeltronic.com/page/pdci-11v.php]
28. Fairchild Semiconductors (24.04.2013) Optoisolaator FOD814 [http://www.farnell.com/datasheets/418834.pdf]
29. PJRC (29.04.2013) FreqMeasure [http://www.pjrc.com/teensy/td_libs_FreqMeasure.html]
30. Texas Instruments (29.04.2013) sagedus-volt muundur LM2907 [http://www.ti.com/lit/ds/symlink/lm2907-n.pdf]
31. Analog Devices (12.05.2013) AD595AQ, temperatuur-volt muundur [http://www.analog.com/static/imported-files/data_sheets/AD594_595.pdf]
32. Kourosh Parsa, Ty A. Lasky, Bahram Ravani (2007). Design and Implementation of a Mechatronic, All-Accelerometer Inertial Measurement Unit.
33. Pei-Chun Lin, Chi-Wei Ho (2009). Design and Implementation of a 9-Axis Inertial Measurement Unit.
34. Tanenhaus, Martin Geis, Tim Carhoun, Dean Holland, Alex (2010). Accurate Real Time Inertial Navigation Device by Application and Processing of Arrays of MEMS Inertial Sensors. IEEE, Position Location and Navigation Symposium (PLANS), 2010 IEEE/ION
35. Forrest M. Mims III (1983) Getting Started in Electronics

Lisad

Lisa 1 – Esmase prototüüpseadme lõplik skeem



Lisa 2 – LinearServo teegi lähtekood – päisefail

```
/*
  LinearServo.h - arduino library for calculating
 circular servo arm motion into linear distance.
 Created by A.Kaver, 2013.
 Released into public domain.
 */
#ifndef LinearServo_h
#define LinearServo_h

#if defined(ARDUINO) && ARDUINO >= 100
#include "Arduino.h"
#else
#include "WProgram.h"
#endif

class LinearServo
{
public:
  LinearServo(int servoArmMinAngle, int servoArmMaxAngle, int servoArmLength);
  int getServoAngleMath(int linearLength);
  int getServoAngleLookup(int linearLength);
  int getServoLinearRange();
  int getServoMinX();
  int getServoMaxX();

private:
  int _servoArmMinAngle;
  int _servoArmMaxAngle;
  int _servoArmLength;
  int _servoMinX;
  int _servoMaxX;
  int _servoLinearRange;
  int _servoLookupTable[256];

  void servoAngleLookupTableFill();

};
#endif
```

Lisa 3 – LinearServo teegi lähtekood – koodifail

```
#if defined(ARDUINO) && ARDUINO >= 100
#include "Arduino.h"
#else
#include "WProgram.h"
#endif

#include "LinearServo.h"

// mathematical constants, used for trigonometry
const float pi = 3.14159265358979323846;
const float degreesToRadians = pi / 180;
const float radiansToDegrees = 180 / pi;

LinearServo::LinearServo(int servoArmMinAngle, int servoArmMaxAngle, int
servoArmLength){
  // set up the servo parameters
  _servoArmMinAngle = servoArmMinAngle;
  _servoArmMaxAngle = servoArmMaxAngle;
  _servoArmLength = servoArmLength;

  // calculate linear movement range acheivable with current servo configuration
  // angles are here in radians
  // this will be positive (right hand from centre)
  _servoMinX = _servoArmLength * cos(_servoArmMinAngle * degreesToRadians);
  // this will be negative (left hand from centre)
  _servoMaxX = _servoArmLength * cos(_servoArmMaxAngle * degreesToRadians);
  // maximum linear movement achievable
  _servoLinearRange = abs(_servoMaxX - _servoMinX);

  //initialize the lookuptable with zeros
  for (int i=0; i<sizeof(_servoLookupTable)/2; i++){
    _servoLookupTable[i] = 0;
  }

  // calculate the lookup table values
  servoAngleLookupTableFill();
}

// calculate the exhaust servo angle - this is slow and expensive
int LinearServo::getServoAngleMath(int linearLength){
  // keep the required length in range
  if (linearLength > _servoLinearRange ){
    linearLength = _servoLinearRange;
  }
  if (linearLength <  0 ){
    linearLength = 0;
  }

  // move the required length into servo X coordinate system
  linearLength = linearLength + _servoMaxX;

  //calculate the required angle
  int res = acos(linearLength / (float) _servoArmLength) * radiansToDegrees;
  return res;
}

// calculate and fill the exhaust servo length to angle conversion table
void LinearServo::servoAngleLookupTableFill(){
  for (int i=0; i <=_servoLinearRange; i++){
    _servoLookupTable[i] = getServoAngleMath(i);
  }
}

// lookup the exhaust servo angle
```

```
int LinearServo::getServoAngleLookup(int linearLength){
  // keep the required length in range
  if (linearLength > _servoLinearRange ){
    linearLength = _servoLinearRange;
  }
  if (linearLength <  0 ){
    linearLength = 0;
  }

  return _servoLookupTable[linearLength];
}

int LinearServo::getServoLinearRange(){
  return _servoLinearRange;
}

int LinearServo::getServoMinX(){
  return _servoMinX;
}
int LinearServo::getServoMaxX(){
  return _servoMaxX;
}
```

Lisa 4 – Mootori tööparameetrite mõõdistamise programmi lähtekood

```
#include <Servo.h>
#include <Wire.h>
#include <LiquidTWI.h>

// enable serial output for debugging
// set to false in production
#define DEBUG true
#define ENABLE_LOG true

// performance counter in debug mode
#if DEBUG==true
unsigned long perfCounter = 0;
#endif

// millis in main loop
unsigned long time = 0;

// ================================================================
#include "LinearServo.h"

// max and min angles have to be tested for real servo,
// usually it is more like 20 to 160
const int servoMin       =   0;  // degrees
const int servoMax       = 180;  // degrees
const int servoArmLength =  30;  // millimeters

LinearServo linearServo(servoMin,servoMax,servoArmLength);

//Exhaust override potentiometer connection
const int EXHAUST_OVERRIDE_PIN=A1;
//Exhaust servo signal connection
const int exhaustServo_Pin = 4;
// servo min angle is achieved with this pulse width
const int exhaustServo_MinPulse = 780;
// servo max angle is achieved with this pulse width
const int exhaustServo_MaxPulse = 2180;

// create a servo object for exhaust length movement
Servo exhaustServo;

// Connect via i2c, default address #0 (A0-A2 not jumpered)
LiquidTWI lcd(0);

// exhaust position
int exhaustPos = 0;
int exhaustOverridePos = 0;
int exhaustServoAngle = 0;
int exhaustServoAnglePrev = 0;

//exhaust temperatures
int leftExhaustTemp = 0;
int rightExhaustTemp = 0;
int leftExhaustTempMax = 0;
int rightExhaustTempMax = 0;

// current RPM value
int rpm = 0;

// throttle position
int throttlePos = 0;
const int throttlePin = A0;

// read analog values, average over X readings
int exhaustOverrideCalculate(){
```

```
  static long sum=0;
  static int count=0;
  static int final = 1023;

  sum = sum + analogRead(EXHAUST_OVERRIDE_PIN);
  count=count+1;
  if (count>=9){
    final = sum / count;
    sum = 0;
    count = 0;
  }

  return final;
}

int exhaustPosOverrideCalculate(int exhaustOverridePos){
  double coef = linearServo.getServoLinearRange() / 1023.0 ;
  int finalPosition = exhaustOverridePos*coef;
  return finalPosition;
}

void updateDisplay(int rpm, int servoPotPos, int throttlePos, int exhaustPos,
int  exhaustServoAngle,  int  leftExhaustTemp,  int  leftExhaustTempMax,  int
rightExhaustTemp, int rightExhaustTempMax){
  String myString = "";

  static unsigned long lastUpdateTime = 0;
  // update display after every X msec
  if (time - lastUpdateTime < 100) {
    return;
  }
  lastUpdateTime = time;

  lcd.setCursor(5,0);
  if (rpm<10) lcd.print("    ");
  else if (rpm<100) lcd.print("   ");
  else if (rpm<1000) lcd.print("  ");
  else if (rpm<10000) lcd.print(" ");
  lcd.print(rpm);

  lcd.setCursor(16,0);
  if (servoPotPos<10) lcd.print("   ");
  else if (servoPotPos<100) lcd.print("  ");
  else if (servoPotPos<1000) lcd.print(" ");
  lcd.print(servoPotPos);

  //throttle
  lcd.setCursor(16,2);
  if (throttlePos<10) lcd.print("   ");
  else if (throttlePos<100) lcd.print("  ");
  else if (throttlePos<1000) lcd.print(" ");
  lcd.print(throttlePos);

  // exhaust length
  lcd.setCursor(7,1);
  if (exhaustPos<10) lcd.print("  ");
  else if (exhaustPos<100) lcd.print(" ");
  lcd.print(exhaustPos);

  // exhaust servo angle
  lcd.setCursor(17,1);
  if (exhaustServoAngle<10) lcd.print("  ");
  else if (exhaustServoAngle<100) lcd.print(" ");
  lcd.print(exhaustServoAngle);


  // exhaust temperatures
  lcd.setCursor(3,2);
```

```
  if (leftExhaustTemp<10) lcd.print("  ");
  else if (leftExhaustTemp<100) lcd.print(" ");
  lcd.print(leftExhaustTemp);

  lcd.setCursor(7,2);
  if (leftExhaustTempMax<10) lcd.print("  ");
  else if (leftExhaustTempMax<100) lcd.print(" ");
  lcd.print(leftExhaustTempMax);


  lcd.setCursor(3,3);
  if (rightExhaustTemp<10) lcd.print("  ");
  else if (rightExhaustTemp<100) lcd.print(" ");
  lcd.print(rightExhaustTemp);
  lcd.setCursor(7,3);
  if (rightExhaustTempMax<10) lcd.print("  ");
  else if (rightExhaustTempMax<100) lcd.print(" ");
  lcd.print(rightExhaustTempMax);

  //time
#if DEBUG==true
  myString = String(time/1000)+"/"+perfCounter;
  perfCounter=0;
#else
  myString = String(time/1000);
#endif
  lcd.setCursor(20 - myString.length(),3);
  lcd.print(myString);
}

int throttleCalculate(){
  static long throttleSum=0;
  static int throttleCount=0;
  static int throttleFinal = 0;

  throttleSum = throttleSum + analogRead(throttlePin);
  throttleCount=throttleCount+1;
  if (throttleCount>=9){
    throttleFinal = throttleSum / throttleCount;
    throttleSum = 0;
    throttleCount = 0;
  }

  return throttleFinal;
}

void writeLog(int throttlePos, int exhaustPos, int exhaustServoAngle){
  static unsigned long lastUpdateTime = 0;
  // update display after every X msec
  if (time - lastUpdateTime < 500) {
    return;
  }
  lastUpdateTime = time;

  Serial.print(time);
  Serial.print(",");
  Serial.print(throttlePos);
  Serial.print(",");
  Serial.print(exhaustPos);
  Serial.print(",");
  Serial.print(exhaustServoAngle);
  Serial.println();
}

void setup(){
#if DEBUG==true
  Serial.begin(115200);
  Serial.println("Setup...");
```

```
#endif

  exhaustServo.attach(exhaustServo_Pin,                    exhaustServo_MinPulse,
exhaustServo_MaxPulse); //attach servo to specified pin

  // set up the LCD's number of rows and columns:
  lcd.begin(20, 4);
  lcd.setBacklight(HIGH);

  // Print a message to the LCD.
  lcd.setCursor(0,0);
  lcd.print(" RPM:XXXXX  POT:XXXX");
  lcd.setCursor(0,1);
  lcd.print("EXHL:  XXX EXHA: XXX");
  lcd.setCursor(0,2);
  lcd.print("LT:XXX/XXX  THT: XXX");
  lcd.setCursor(0,3);
  lcd.print("RT:XXX/XXX");

#if DEBUG==true
  Serial.println("Setup done!");
#endif
}

void loop(){
  // global timetick
  time = millis();

#if DEBUG==true
  perfCounter++;
#endif

  throttlePos = throttleCalculate();
  throttlePos = map(throttlePos, 4, 1021, 0, 1023);

  //this is for engine bench testing, using sliding potentiometer
  // for exhaust position
  exhaustOverridePos = exhaustOverrideCalculate();
  exhaustOverridePos = map(exhaustOverridePos, 4, 1021, 0, 1023);

  // convert the exhaustOverridePos into millimeters
  exhaustPos       =       map(exhaustOverridePos,       0,       1023,       0,
linearServo.getServoLinearRange());
  // get the required servo angle
  exhaustServoAngle = linearServo.getServoAngleLookup(exhaustPos);
  exhaustServoAngle = map(exhaustServoAngle,servoMin,servoMax,0,180);

  if (exhaustServoAngle!=exhaustServoAnglePrev){
    exhaustServoAnglePrev = exhaustServoAngle;
    exhaustServo.write((byte) exhaustServoAngle);
  }


  updateDisplay(rpm,      exhaustOverridePos,      throttlePos,      exhaustPos,
exhaustServoAngle,   leftExhaustTemp,   leftExhaustTempMax,     rightExhaustTemp,
rightExhaustTempMax);

#if ENABLE_LOG==true
#if DEBUG==true
  writeLog(exhaustOverridePos, exhaustPos, exhaustServoAngle);
#endif
#endif

}
```