TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Laur Kaarel Leoke 155715

TAHHOMEETER VANEMALE AUTOLE

Bakalaureusetöö

Juhendaja: Andres Rähni
Tehnikateaduste magister

Tallinn 2020

Autorideklaratsioon

Kinnitan, et olen koostanud antud lõputöö iseseisvalt ning seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on töös viidatud.

Autor: Laur Kaarel Leoke

15.01.2020

Annotatsioon

Selle bakalaureusetöö eesmärgiks on luua Arduino mikrokontrollerit kasutades tahhomeeter jagajaga süütesüsteemiga autole. Töö tulemus võib huvi pakkuda autoomanikele, kelle autos ei ole tahhomeetrit või soovivad autosse digitaalse näidikuga tahhomeetrit.

Töös esmalt kirjeldatakse näidikute valmistamiseks valitud komponente ja selgitatakse, kuidas saab jagajaga süütesüsteemist kätte pöörete lugemiseks vajaliku signaali. Seejärel kirjeldatakse komponentide valmimist ning juhtloogikat.

Töö tulemuseks on reaalselt töötav tahhomeeter, mida saab kasutada auto mootori pöörlemiskiiruse jälgimiseks.

Lõputöö on kirjutatud Eesti keeles ning sisaldab teksti 23 leheküljel, 7 peatükki, 8 joonist, 0 tabelit.

Abstract

Tachometer for older cars

The purpose of this thesis is to create a tachometer for cars that use a distributor based ignition system. The product of the work could be of interest to owners of cars that do not have a tachometer or want to have a tachometer with digital display.

This thesis first describes the components used in the making of the display and explains how a signal for measuring revolutions of the engine is acquired from the ignition system. It then describes the manufacture of the components and the control logic.

The result of this thesis is a working tachometer that can be used for observing the engine speed of a car.

The thesis is in Estonian and contains 23 pages of text, 7 chapters, 8 figures, 0 tables.

Lühendite ja mõistete sõnastik

LED	Light emitting diode
SSD	Seven segment display
GND	Maandus / 0 voldi siin
VCC	Toide / 5 voldi siin
DIO	Digital in/out - digitaalne sisend väljund
CLK	Clock - kellasignaal
DIN	Data in - andmed sisse
IP	*Ingress protection*
IEC	*International Electrotechnical Commission*

Sisukord

Jooniste loetelu

1 Sissejuhatus

Tänapäeval on peaaegu igal uuel sisepõlemismootoriga autol olemas tahhomeeter, mille abil saab juht kiire ülevaate auto mootori hetkepööretest. Tahhomeeter on eriti oluline manuaal käigukastiga autodes, sest manuaalkäigukastiga peab autojuht ise otsustama milline käik on hetke olukorras parim ja ise valima selle käigu. Tahhomeeter aitab autojuhil teha seda valikut. Ilma tahhomeetrita on väga lihtne valida ebasobiv käik ja selle tõttu kulutada liigselt kütust või isegi vigastada auto mootorit. Näiteks maanteele keerates tuleb saavutada vastav sõidukiirus ja selle saavutamiseks on vaja suurendada mootori pöördeid. Ilma tahhomeetrita ei tea autojuht kui kiiresti mootor töötab ning võib vahetada käiku liiga vara ja kiirendada aeglasemalt kui vaja või hoopis oodata liiga kaua käiguvahetusega ning kõige halvemal juhul vigastada mootorit liiga kõrgete pööretega. Paljudel vanematel autodel ei tulnud tehasest kaasa tahhomeetrit või see oli lisavarustuse nimekirjas ja auto ostja pidi selle eest juurde maksma.

Töö ülesandeks on püstitatud digitaalse tahhomeetri valmistamine jagajat kasutava süütesüsteemiga autole ning järgnevates peatükkides on välja toodud kasutatud komponendid, antakse ülevaade kuidas töötab auto süütesüsteem, tuuakse välja tekkinud probleemid ja leitud lahendused ning tutvustatakse keskkonnatingimuste vastu kasutatuid meetmeid.

Süütesüsteemi tutvustamine on oluline signaali allikast aru saamiseks ja optroni ülesande selgitamiseks. Komponentide osas tutvutakse kasutatud komponentide omadusi. Teostuse osas on välja toodud vaheetapid töö edenemisest ning ülevaade kasutatavast juhtloogikast. Lahenduste testimises on kirjeldatud tekkinud probleeme. Keskkonnatingimuste vastu kaitsmine toob välja töökindluse tagamiseks kasutatud meetmed esinevates keskkonnatingimustes.

2 Komponendid

Antud peatükis antakse ülevaade töös kasuatud komponentidest ja nende *library*-test ning seletus miks valiti just need komponendid.

2.1 Arduino Uno

Arduino Uno on üks kõige levinumaid arendusplaate hobielektroonikute ja prototüüpide valmistajate seas just oma hea hinna ja väga laialdase dokumentatsiooni pärast. Arduinol on programmide kirjutamiseks *open-source* arenduskeskkond Arduino IDE [10] ja programme kirjutatakse C/C++ keeles [9].

Antud töös valiti Arduino Uno põhiliselt selle tõttu, et see oli töö autoril juba olemas ja töö autoril oli olemas varasem kogemus Arduinot kasutavate elektroonikaprojektidega. Lisaks Arduino on võimeline töötama ka paljudes eri töötingimustes.

2.2 MAX7221

MAX7221 on väike *display* draiver mis on mõeldud kuni kaheksa numbriga SSD või kuni 64 LED-iga maatriksi juhtimiseks ning see kasutab kasutab serial sisend/väljundit [11]. MAX7221 kasutab LED maatriksi juhtimiseks ühise katoodi meetodit st. LED maatriksis on igal 8-st LEDist koosneval grupil ühine katood [6]. Kuni 16 MAX7221 on võimalik üksteisega ühendada. MAX 7221 on 5 sisendit(VCC, GND, CLK, DIN, LOAD). MAX7221 juhtimiseks kasutatakse *LedControl library*-t [3].

MAX7221 valiti põhiliselt 64 LED-i juhtimise võime poolest, sest töö alguses ei olnud teada mitu LED-i on tarvis kasutada optimaalseima näidiku valmistamiseks.

2.3 TM1637 SSD moodul

TM1637 poolt juhitud 4 numbriga SSD moodul, mis kasutab serial sisend/väljundit. Moodulil on 4 sisend kontakti (GND, VCC, DIO, CLK). TM1637 SSD mooduli juhtimiseks kasutati TM1637 *library*-t [4].

Otsustati valida TM1637 SSD moodul sellepärast, et see on juba valmis moodul ning sellel on väga hea Arduino *library.* Lisaks on TM1637 ühendamiseks vaja ainult 4 juhet.

2.4 LM2596 DC-DC muundur

LM2596 DC-DC muundur teeb auto toitesüsteemi pingest (12-14 volti) Arduinole sobiva 5 volti.

Esialgu oli plaanis kasutada vana auto telefonilaadijat, mis toimis, aga katsetamise käigus selgus, et telefonilaadija pakutud toide oli väga mürarikas ja kindluse mõttes vahetati see välja LM2596 DC-DC muunduri vastu.

2.5 H11M1

H11M1 on sisse ehitatud Schmidti trigeriga optron ja mõeldud kõrgete sagedustega signaalide puhul kasutamiseks [5] [12]. Sisse ehitatud Schmidti triger aitab kaasa müra summutamisele sisendsignaalis.

H11M1 optroni kasutati Arduino ja auto süütesignaali isoleerimiseks, et vältida vigastusi Arduinole ja teistele 5 voldise tööpingega komponentidele. H11M1 valiti põhiliselt sisse ehitatud Schmidti trigeri pärast teadmisega kui mürarikas võib auto süütesüsteem olla ja taheti kasutada võimalikult palju müra filtreerivaid komponente.

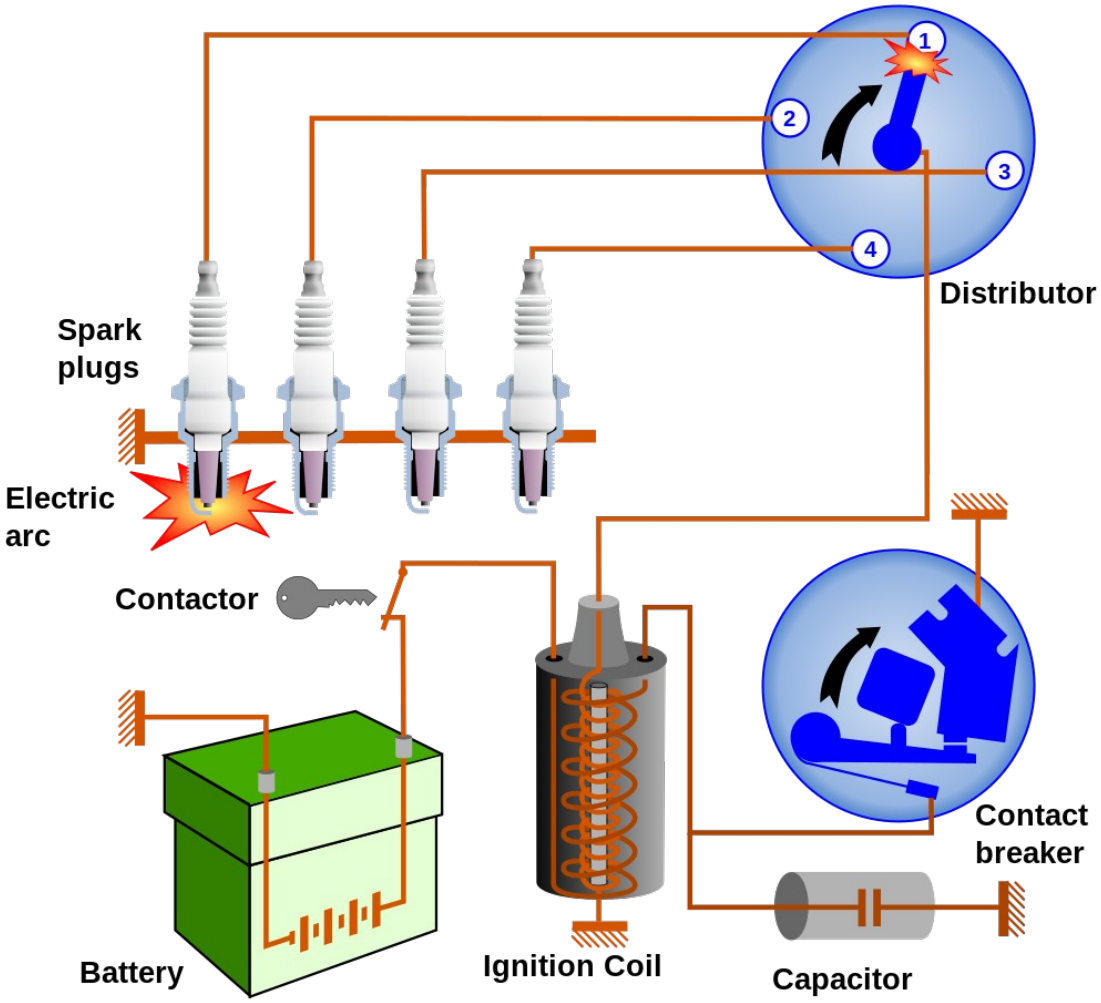
3 Ülevaade auto süütesüsteemist

Käesolevas peatükis kirjeldatakse auto süütesüsteemi toimimist, sest sellest süsteemist saab hinnangu mootori pöörlemiskiiruse kohta.

Tavalistel kasutavad autod oma süsteemides 12 voldist pinget, mis tegelikul on 13-15 volti auto töötamise ajal. Vanematel autodel koosneb süütesüsteem akust (*battery*), süütepoolist (*ignition coil*), jagajast (*distributor*) ja süüteküünaldest (*spark plugs*) ning seda kutsutakse jagajaga süüteks (vt. Joonis 1).

Süütepool on vajalik aku 12 voldise pinge suurendamiseks süüteküünaldele sädeme tekitamiseks vajalikuks pingeks, mis ulatub kümnete tuhandete voltideni. Nii kõrge pinge tekitamiseks on süütepooli 12 voldi mähise üks ots ühendatud läbi süüteluku aku positiivse terminali külge ja teine ots on ühendatud läbi spetsiaalse lüliti aku negatiivse terminaliga. Spetsiaalne lüliti (*contact breaker*) asub jagaja küljes ja ühendab süütepooli akust lahti iga silindri jõudmisel ülemisse surnud seisu tekitades süütepooli sekundaarses mähises kümnetesse tuhandetesse voltidesse ulatuva pinge mida kasutatakse silindrites õhu ja kütuse segu süütamiseks [8].

Selleks, et säde tekkiks õiges silindris on jagaja. Jagaja koosneb omakorda rootorist ja jagaja kaane küljes olevatest klemmidest ning on tavaliselt ühendatud auto nukkvõlliga, sest neljataktilise tööpõhimõttega mootori jagaja peab olema mootoriga sünkroonis ja pöörlema mootorist poole aeglasemalt. Jagaja rootori pöörlemisega liigub rootor jagaja kaane sees olevate klemmide vahel ja igale klemmile vastab juhtmega ühendatud süüteküünal. Kui rootor on klemmiga kohakuti suunatakse süütepooli poolt tekitatud pinge süüteküünlasse ja tekib sädelahendus süüteküünla elektroodide vahel, mis süütab silindris kütuse ja õhu segu [8]. Jagajaga süütesüsteem toodab igal pöördel mootori silindrite arvust poole vähem süüteimpulsse.



Joonis 1. Süütesüsteemi joonis [8].

Tänapäeval on rohkem levinud *coil-on-plug* lahendus, mis koosneb väntvõlli ja nukkvõlli positsiooni andutitest ja igal süüteküünlal asetsevast süütepoolist. Kogu süüte jagamist juhib auto mootori juhtelektroonika saades infot mootori oleku kohta anduritest. Vastavalt mootori olekule jagab mootori juhtelektroonika igale süütepoolile eraldi signaale sädeme tekitamiseks [8].

Antud töö lahendus saab signaali pöörete lugemiseks süütepooli 12 voldi mähise pealt ja igale mootori pöördele vastab kaks süüteimpulssi.

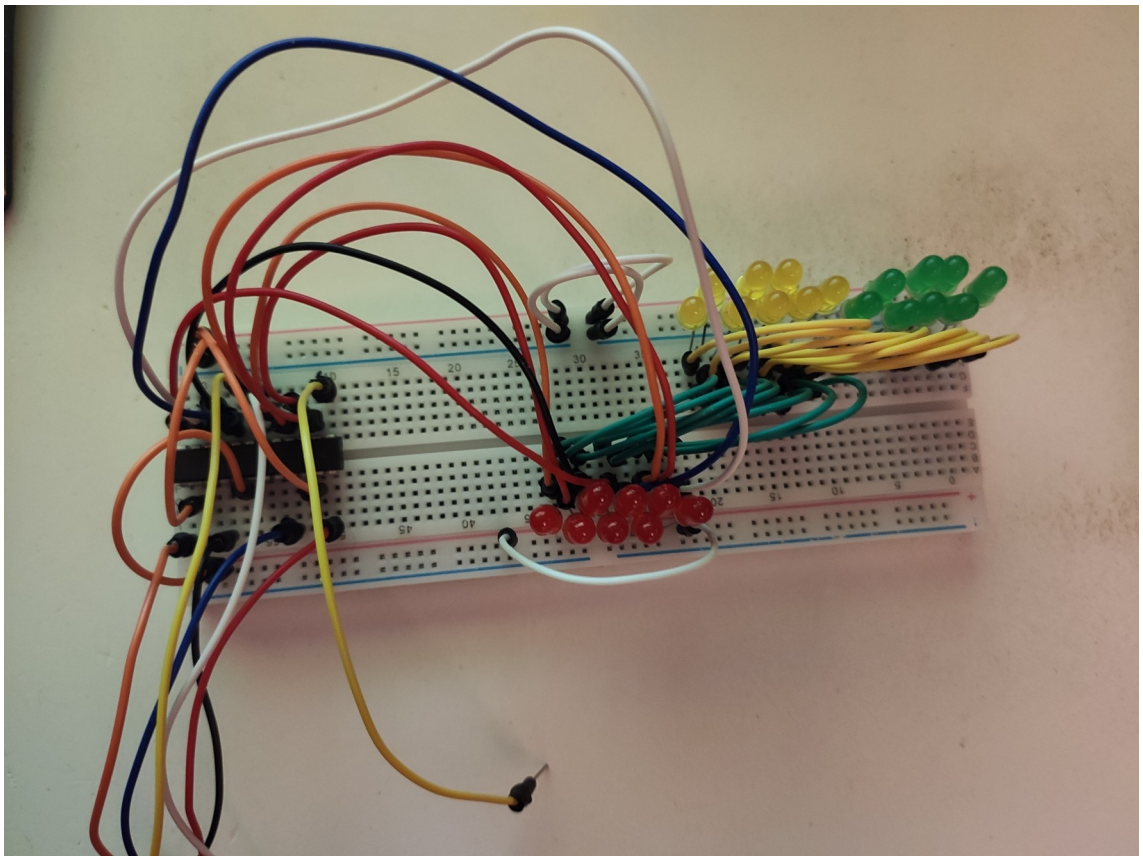
4 Teostus

Lõputöö käigus valmis Arduino kontrollerit kasutades digitaalne tahhomeeter, mis on mõeldud vanemat tüüpi auto süütesüsteemiga töötama. Selles peatükis on tutvustatud lõputöö töö tehnilist teostust. Näidikuks valiti 24 LED-i ja igale LED-ile vastab 250 pööret minutis ja LED-idega näidatavaks maksimaalseks pöörete arvuks on 6000 pööret minutis, mis on parajasti paljude 60-80-ndate sõiduautode mootorite maksimumpöörded. Mootoripöörete täpsemaks näitamiseks lisati SSD, mis näitab mootoripööreid täisarvuna ja ilma 6000 pöörde piiranguta.

4.1 Komponentide ja *library*-tega tutvumine

Tööd alustati tulevaste näidikute riistvaraga tutvudes, ning esimese asjana saadi tööle maketeerimisplaadil TM1637 juhitud SSD moodul. SSD moodulil on TM1637 *library*-ga [4] võimalik juhtida iga segmenti eraldi ja luua eelnevalt kirjeldatud olekuid ning neid hiljem esile kutsuda. Veel on võimalik sisestada SSD-le ka lihtsalt numbriline väärtus, mida näidata ning selles töös ongi vaja näidata numbrilist väärtust.

Järgmisena tehti eraldi maketeerimisplaadile MAX7221 ühendused LED-ide ja Arduinoga [1]. MAX7221 jaoks kasutatud *LedControl library*-ga [3] on võimalik juhtida ka mitut omavahel ühendatud MAX7221 LED draiverit aga antud töös piirduti ühega. MAX7221 kasutati 24-st LED-ist koosneva maatriksi juhtimiseks. Maatriks oli jaotatud kolmeks kaheksast LED-ist koosnevaks reaks (vt. Joonis 2 ja Lisa 2). Töös juhiti LED maatriksit *LedControl library* rea juhtimise funktsiooniga. Võimalik on veel juhtida iga LED-i individuaalselt või tulba kaupa. Tulba kaupa juhtimine oli dokumentatsiooni järgi kõige aeglasem juhtimise meetod ja valitud rea juhtimine kõige kiirem (vt. Lisa 1 Joonis 8).



Joonis 2. MAX7221 moodul maketeerimisplaadil.

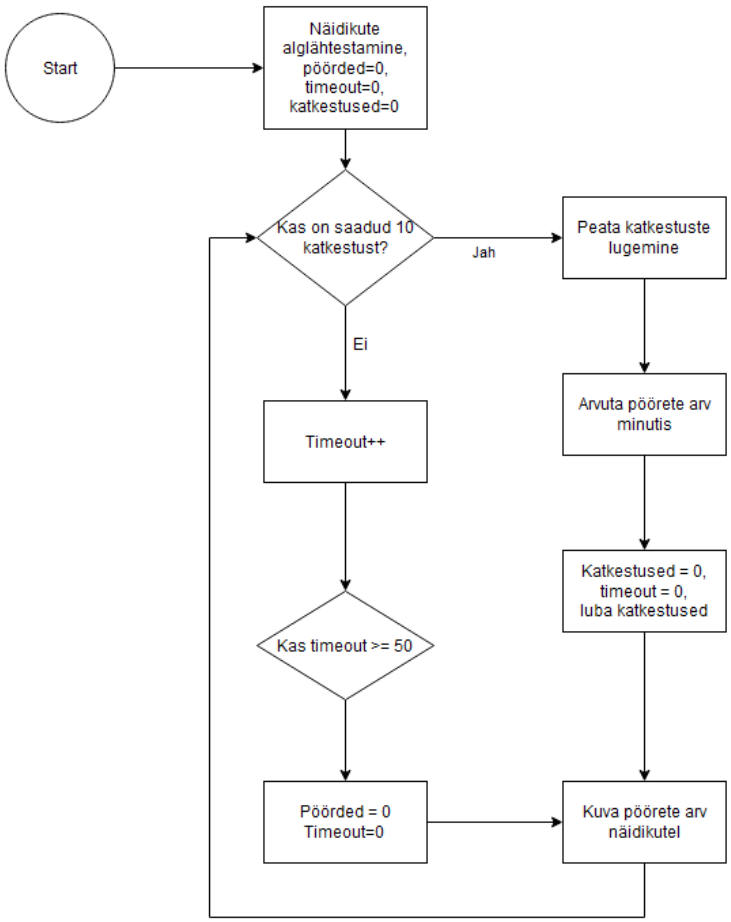
Töö dünaamilisemaks katsetamiseks lisati maketeerimisplaadile potentsiomeeter, mille abil simuleeriti pöörete arvu. Potentsiomeetriga katsetusi tehes selgus, et esialgne viis andmete näitamisel MAX7221 juhitud näidikul oli ühe veaga. Viga seisnes selles, et kui väärtus hüppas üle terve kaheksast ledist koosneva rea, siis jäid need ledid oma endisesse seisu. Ledide endisesse seisu jäämise vältimiseks tuli lisada kontroll juhuks, kui signaal on hüpanud üle kaheksast ledist koosneva rea ja siis kustutada või süüdata see rida vastavalt.

Peale näidikute tööle saamist potentsiomeetri poolt sisestatud väärtusega hakati uurima võimalusi pulssidega signaali sisselugemiseks.

Autost tuleva signaali vastuvõtmiseks otsustati antud töös kasutada Arduino katkestusi [2]. Katkestuste kasutamine sisse tuleva signaali lugemiseks valiti põhjusega, et katkestusi loetakse ka teiste funktsioonide töötamise ajal ja ei teki näidikute töös paigalseismise hetki, kui programm parasjagu sissetulevat signaali lugema jääb.

Katkestuste katsetamiseks lisati maketeerimisplaadile nupp, mida kiirelt vajutades simuleeriti auto süütesüsteemist tulevat signaali. Nupu kasutamine katkestuste katsetamiseks ei osutunud optimaalseks ning see asendati signaaligeneraatoriga.

4.2 Tahhomeetri juhtalgoritm



Joonis 3. Tahhomeetri programmi vooskeem.

Tahhomeetri eesmärgiks on näidata auto kasutajale mootori hetkepööredid. Mootori hetkepöörete arvutamiseks kasutab Arduinol jooksev programm kahe arvutuse vahelist aega ja kokkuloetud katkestuste arvu ning teisendab selle minutiteks (vt . Lisa 1 Joonis 8). Tahhomeetri algoritm on üles ehitatud kindla katkestuste arvu järel arvutama aega, mis kulus katkestuste lugemiseks (vt. Joonis 3). Hetkel on arvutuste vahele valitud 10

katkestust. Arvutuste vahelise katkestuste koguse suurendamine muudab tahhomeetri täpsemaks aga samas vähendab see tahhomeetril näidatud väärtuse uuendamise kiirust.

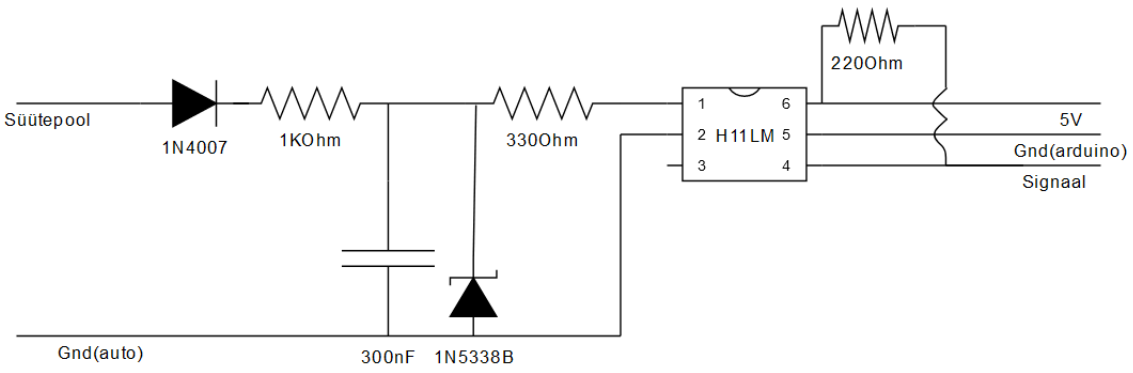
Valitud loogika pärast on mootori madalamate pöörete lugemisel tahhomeetri näidikute uuendamiskiirus aeglasem kuid kasvab koos mootoripöörete kasvamisega. Mootori seiskumise kontrolliks valiti 50 koodi tsüklit, mis piirab praeguse koodi (vt. Lisa 3) pikkusega tahhomeetri minimaalselt näidatavad pöörded 500 ringi. Seda piirangut saab viia madalamate mootori pöörete peale suurendades sisendsignaali vahele jäävate koodi tsüklite kogust, kuid sellega langeb kiirus millega tahhomeeter end nullib.

Tahhomeetri planeeritud kasutatavust 500 pöördene alampiir ei häiri, sest on väga vähe autodes kasutatavaid sisepõlemismootoreid mis on võimelised töötama madalama mootorikiirusega kui 500 pööret minutis. Mootorid, mis on võimelised töötama madalamate pööretega on tavaliselt rohkemate silindritega ja seetõttu tekitavad rohkem süüteimpulsse, sest jagajaga süütesüsteem toodab igal pöördel mootori silindrite arvust poole vähem süüteimpulsse ja silindrite arvu suurendamisega suurendati ka süüteimpulsside arvu pöörde kohta.

4.3 Moodulite valmistamine

Töö järgmiseks etapiks oli komponentidest makettplaatidele moodulite tegemine.

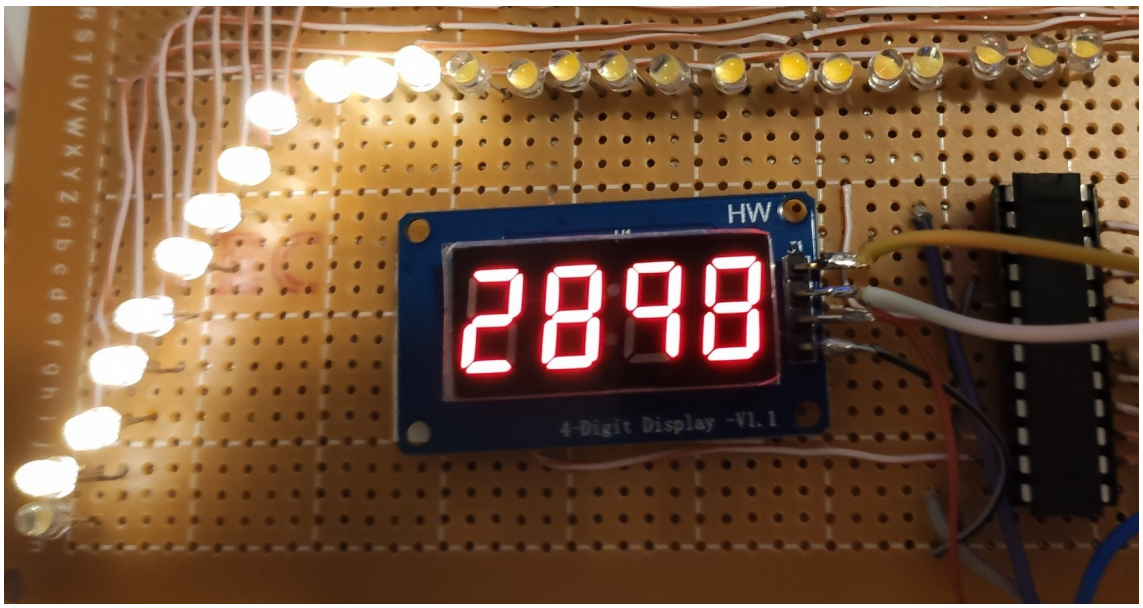
Maketeerimisplaadil ei katsetatud ainukese moodulina optroni moodulit, sest see ei vajanud tarkvara poolt juhtimist ja koosnes vähestest komponentidest ning oli vajadusel kergesti modifitseeritav.



Joonis 4. Optroni mooduli skeem.

Optroni moodul koosneb süütesüsteemi poolel kahest takistist, mis vähendavad voolu H11M1 optroni jaoks sobivaks, ühest dioodist signaali negatiivse poole blokeerimiseks, ühest 5.1 voldisest Zeneri dioodist ja kondensaatorist müra summutamiseks. Arduino poolel on VCC ja signaali juhtme vahel *pull-up* takisti. Süütesüsteemi ja Arduino poolte vahel on H11M1 optron (vt. Joonis 4).

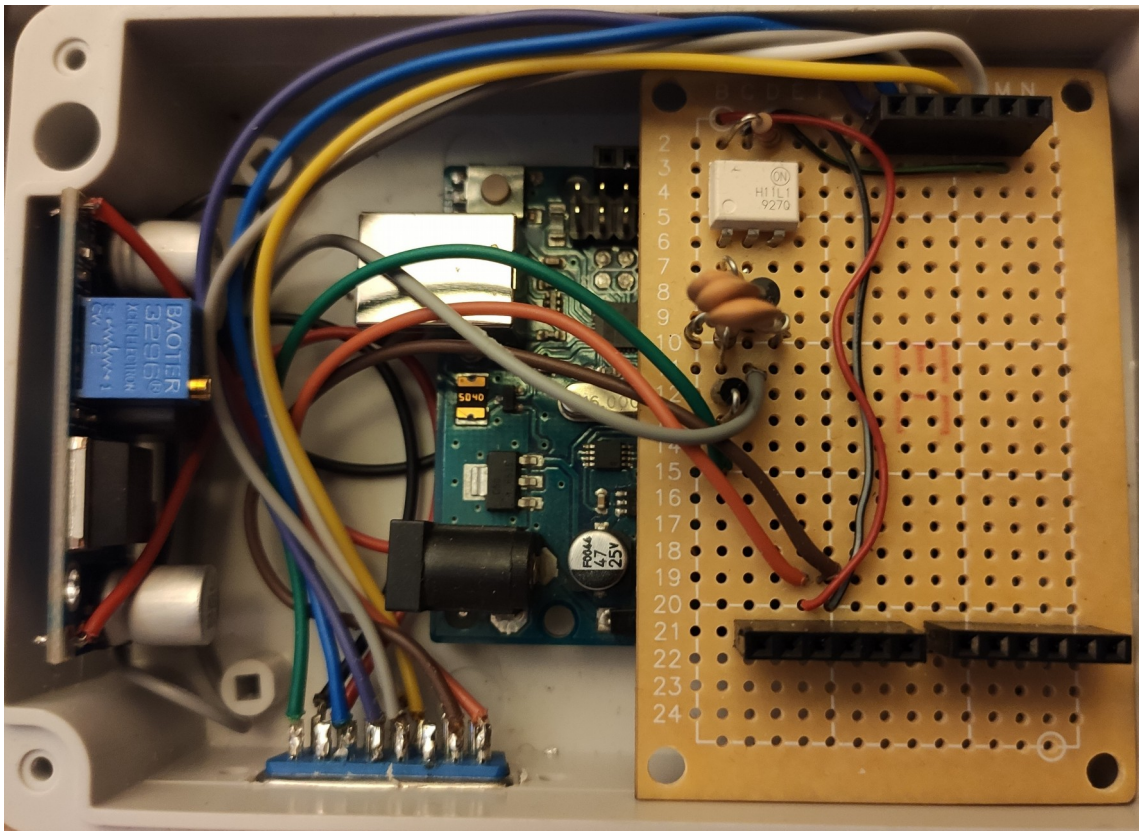
Peale optroni mooduli valmimist joodeti kokku MAX7221 poolt juhitud LED näidik ning lisati TM1637 poolt juhitud SSD moodul (vt. Joonis 5).



Joonis 5. MAX7221 ja TM1637 moodul signaaligeneraatorist saadud andmeid näitamas.

Peale moodulite valmimist hakati ette valmistama karpi Arduino, optroni ja toitemooduli jaoks. Karbi kõige suuremaks ülesandeks saab niiskuse tõrjumine. Karbiks valiti universaalne tihendiga plastkarp, mis on just piisavalt suur, et mahutada ära Arduino, optroni moodul ja toitemoodul ning mille ühte külge tehti avaus pistiku jaoks (vt. Joonis 6).

Töös otsustati kasutada karbi ja näidikumooduli vahel DA-15 pistikut. DA-15 pistikus on 15 klemmi kahel real ja hetkel on neist kasutatud 10 jättes 5 vabaks võimalikele lisa näidikutele ja anduritele. Pistiku kasutamine võimaldab eemaldada väga vähese vaevaga karbi, milles on Arduino ning jätta näidikumoodul autosse.



Joonis 6. Optroni moodul ja toitemoodul Arduino külge kinnitatuna

5 Lahenduste testimine

Tahhomeetri katsetamisel Moskvitch 2140 peal selgus, et süütepoolist tulev signaal on mürarikkam kui oodatud ja selle tõttu näitas tahhomeeter mootori tühikäigul 700-800 pöörde asemel 2000. Ostsilloskoobiga optroni väljundit uurides oli näha väikest sakki peale põhiimpulssi. Müra paremaks filtreerimiseks oli vaja kasutada suuremat kondensaatorit optroni sisendis. Peale suurema kondensaatori (vt. Joonis 4) lisamist hakkas tahhomeeter normaalseid tühikäigu pöördeid näitama ja ostsilloskoobiga vaadates oli optroni väljundsignaal palju puhtam.

Teiseks probleemiks päris autoga katsetades sai olema MAX7221 draiver. Katsetuste alguses töötas LED näidik perfektselt aga ühel hetkel süttisid kõik LED-id täisvõimsusel ja hetke pärast kustusid. Peale Arduino taaskäivitamist jätkus sarnane käitumine. Ostsilloskoobiga signaale vaadates ei olnud võimalik leida viga, kuid signaalide kontrolli käigus sai kannatada üks MAX7221 draiver ja Arduinol lõpetas töötamise üks väljund. Kahjustuste põhjuseks võis olla läbi ostsilloskoobi sondi tulnud 12 volti pinget. Täpsemalt MAX7221 dokumentatsiooni lugedes selgus, et Arduino ja MAX7221 vahelised juhtmed olid liiga pikad ja lisa segajana võis käituda signaalijuhe süütepoolilt, mis jooksis MAX7221 ja Arduino vaheliste juhtmete kõrval.

Peale MAX7221 ja Arduino ATmega328P välja vahetamist uute vastu ning juhtmete lühendamist hakkas LED näidik uuesti ootuspäraselt tööle.

6 Keskkonnatingimuste vastu kaitsmine

Üheks suureks probleemiks autosse elektroonika tegemisel on autos olev keskkond. Eesti kliima muudab autod seest niiskeks ja niiskus on üks suuremaid elektroonika kahjustajaid ning just autosse elektroonika tegemisel tuleb mõelda kuidas kaitsta elektroonikat niiskuse eest. Teiseks suureks probleemiks eriti talvel on temperatuur. Talvel jahtuvad autod miinuskraadide käes seistes maha aga kui külmas seisnud auto käima panna siis elektroonika peab endiselt töötama laitmatult. Samas suvel päikese käes seisvas autos võib temperatuur tõusta väga kõrgele ja ohustada elektroonikat ülekuumenemisega.

Lahendusi nendele probleemidele on mitmeid aga otsustati kasutada Arduino, optroni mooduli ja toitemooduli kaitseks niiskuskindlat karpi. Temperatuuri probleemile leiti lahendus sobivate komponentide kasutamise kaudu. Tahhomeetri valmistamiseks valiti komponendid mille ettenähtud töötemperatuur oleks -40°C kuni 85°C.

Suviste kõrgete temperatuuride vastu kaitseks kõige lihtsam lahendus on elektroonika varjamine otsese päikesevalguse eest paigaldades tahhomeetri karp Arduinoga armatuuri alla ning saab lisada karpi ka temperatuurianduri, mille abil saab hoiatada kasutajat ülekuumenemisest.

Vee ja tolmukindlust tähistatakse elektroonikaseadmete puhul IP-koodiga. IP-kood on määratud IEC standardiga 60529 [7] koosneb tähisest IP ja kahest numbrist nt. IP-22. IP-koodi esimene number näitab kui kaitstud on seade tahkete osakeste vastu ja teine number näitab veekindlust. Parim IP-kood mis ei sisalda survestatud veejugasi on IP-68, mis tähendab, et seade on täielikult tolmukindel ja peab vastu kuni kolme meetri sügavusel vee all üle 30 minuti. On ka IP-koode, mille saamiseks peab toode vastu pidama survestatud veejugadele ja kõrge temperatuuriga veejugadele [7].

Selle töö jaoks on sobivad IP-koodid IP-54 ja IP-64 mis tähendab, et toode on tolmu vastu kindlustatud või täiesti tolmukindel ning suvalisest küljest pritsiv vesi ei kahjusta seadet. See on antud rakendusele piisav, sest ei ole ette nähtud vee all töötamist küll aga

võib vanema, ilma salongifiltrita auto sisemuses olla päris palju tolmu. Karp kuhu Arduino paigutati oli algselt IP-68 koodiga, aga pistiku jaoks ava tegemine vähendas vee ja tolmukindlust.

7 Kokkuvõte

Töö eesmärk oli luua tahhomeeter vanemat tüüpi süütesüsteemiga autole. Loodud prototüüp võimaldab kaasajastada vanemad autod, aitab juhtidel vältida ebaoptimaalseid mootoripöördeid ja kahjustusi auto mootorile.

Töö esimeses osas anti ülevaade kasutatud komponentidest ja auto süütesüsteemi toimimisest ning töö teises osas on kirjeldatud komponentide kasutamist, juhtloogikat ja teostusel esile kerkinud raskustest ja neile leitud lahendustest.

Töö koostamisel õppis autor reaalsest keskkonnast saadud analoogsignaali töötlemist mikrokontrolleri jaoks sobivale kujule ning saadud signaali töötlemist kasutajate jaoks kergesti arusaadavale kujule. Veel oli oluline osa elektroonika komponentide montaažil jootmisoskuse parandamisel ning saadi kogemust sardsüsteemide koostamisel.

Töö tulemuseks on reaalne tahhomeeter, mida saab kasutada autos mootoripöörete jälgimiseks, ning millele on võimalik lisada näidikuid ja andureid lisafunktsioonide jaoks.

Töö edasiseks arendusteks saab SSD-d kasutada lisatud andurite andmete või piiratud tähtedega hoiatussõnumite näitamiseks. Saab kasutada ka pistikul tühjaks jäänud 5 klemmi andurite lisamiseks ning kombineerida lisatud andurite andmeid MAX7221 daiveril kasutamata 40 LED-i juhtimise võimekusega näiteks mootoritemperatuuri või kütuse koguse näitamiseks.

Kasutatud kirjandus

[1] “Wiring and schematics” [Online] https://playground.arduino.cc/Main/MAX72XXHardware/ (27.11.2019)

[2] “Using interrupts” [Online] https://www.arduino.cc/reference/en/language/functions/external-interrupts/attachinterrupt/ (05.12.2019)

[3] E. Fahle, “Examples/LCDemoMatrix/” [Online] https://github.com/wayoda/LedControl/ (27.11.2019)

[4] Avishay, “Examples/TM1637Test/” [Online] https://github.com/avishorp/TM1637 (27.11.2019)

[5] “Description” [Online] https://www.onsemi.com/pub/Collateral/H11L3M-D.PDF (14.11.2019)

[6] “General Description” [Online] https://datasheets.maximintegrated.com/en/ds/MAX7219-MAX7221.pdf (27.11.2019)

[7] “IP Code”, “Code breakdown”[Online] https://en.wikipedia.org/wiki/IP_Code (04.12.2019)

[8] “Mechanically timed ignition” [Online] https://en.wikipedia.org/wiki/Ignition_system (14.11.2019)

[9] “What is Arduino” [Online] https://www.arduino.cc/ (14.01.2019)

[10] “Arduino 1.8.10” [Online] https://www.arduino.cc/en/main/software (14.11.2019)

[11] “Overview” [Online] https://www.maximintegrated.com/en/products/power/display-power-control/MAX7221.html (14.11.2019)

[12] “Features” [Online] https://www.onsemi.com/products/optoelectronics/high-performance-optocouplers/high-speed-logic-gate-optocouplers/h11l1m (14.11.2019)

Lisa 1 – Arduino kood

```
attachInterrupt(0, get_u, RISING);//interruptide aktiveerimine
u = 0;
rpm = 0;
timeold = 0;//andmete algväärtustamine
void loop() {
//siin vahepeal on andmete näitamise kood
  if (u >= 10) {
     //uuendab pöörete arvu displayl iga 10 signaali tagant
     noInterrupts();//interruptid välja arvutuse ajaks
     rpm = (u*60000/(millis() - timeold))/2;//60000 millis=1min
      //pulsside arv korrutatakse suureks et vältida koguväärtuse
ümardamist 0ks, sest andmetüüpe mida valemis kasutatakse on palju
     timeold = millis();
     interrupts();//interruptid tagasi
     u = 0;
     timeout = 0;
      }else{
                timeout++;
      }
      if (timeout>50) rpm = 0, timeout=0;
    }//vajalik pöörete nullimiseks juhul kui mootor seiskub
}
void get_u(){
      u++;   //iga mootori pöördega tekkib 2 interrupti
}
```

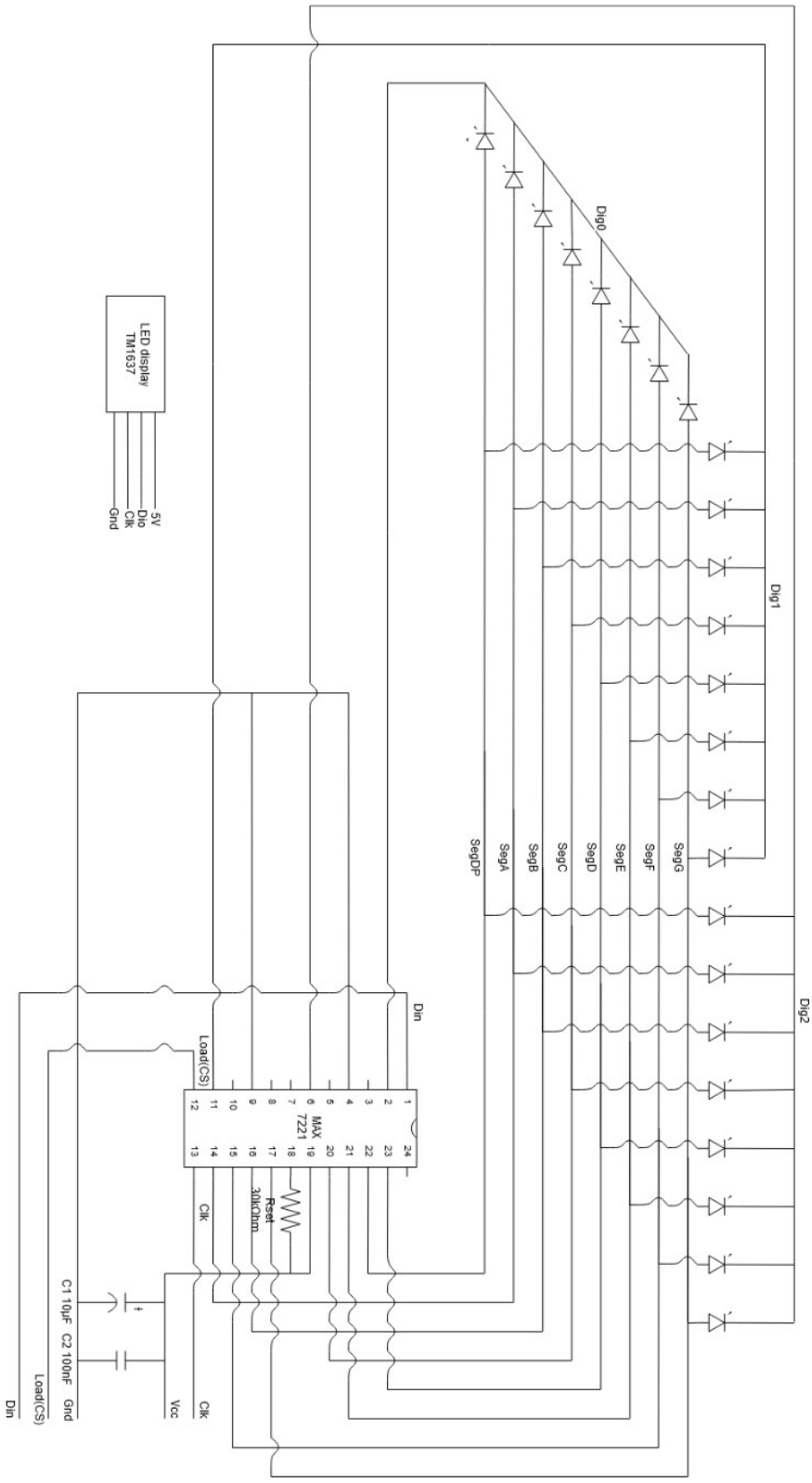
Joonis 7. Pöörete arvutamise kood.

```
      //3. moodul
      if(i>1750&&i<2000)lc.setRow(0,0,B11111110); else
if(i<1750&&i>1500) lc.setRow(0,0,B11111100);
      if(i>1500&&i<1750)lc.setRow(0,0,B11111100); else
if(i<1500&&i>1250) lc.setRow(0,0,B11111000);
      if(i>1250&&i<1500)lc.setRow(0,0,B11111000); else
if(i<1250&&i>1000) lc.setRow(0,0,B11110000);
      if(i>1000&&i<1250)lc.setRow(0,0,B11110000); else
if(i<1000&&i>750) lc.setRow(0,0,B11100000);
      if(i>750&&i<1000)lc.setRow(0,0,B11100000); else if(i<750&&i>500)
lc.setRow(0,0,B11000000);
      if(i>500&&i<250)lc.setRow(0,0,B11000000); else if(i<500&&i>250)
lc.setRow(0,0,B10000000);
      if(i>250&&i<0)lc.setRow(0,0,B10000000); else if(i<250)
lc.setRow(0,0,B00000000);


      if(i>2000)lc.setRow(0,0,B11111111); else if(i<250)
lc.setRow(0,0,B00000000);
      if(i>4000)lc.setRow(0,1,B11111111); else if(i<2250)
lc.setRow(0,1,B00000000);
      if(i>6000)lc.setRow(0,2,B11111111); else if(i<4250)
lc.setRow(0,2,B00000000);
      //Vajalik olukordades, kus pöörete arv hüppab ühelt väärtuselt
teisele nii kiiresti, et jätab ühe terve 8 LED mooduli vahele ja selle
tõttu jäävad ledid kas põlema või kustu
```

Joonis 8. Näide LED mooduli juhtimisest.

Lisa 2 – LED mooduli skeem



Lisa 3 – Kogu programmi kood

```
  // Include the library:
    #include <TM1637Display.h>
    #define CLK 4 //7seg disp
    #define DIO 3 //7seg disp
    #include "LedControl.h"
    #include <binary.h>
    LedControl lc=LedControl(7,6,5,1); //pin 7 on data in, pin 6 on
clk, pin 4 on load 1 on mitu max7221 on ühendatud

    // Create display object of type TM1637Display:
    TM1637Display display = TM1637Display(CLK, DIO);
    int i;
    volatile byte u; //sisend signaal
     unsigned int rpm;//unsigned int
     unsigned long timeold;
      int timeout = 0;
    void setup() {
      // 7seg display tühjaks
      display.clear();
      //äratab max7221 üles unereziimist
      lc.shutdown(0,false);
      //max7221 LEDide heleduse määramiseks väärtused 0 kuni F
      lc.setIntensity(0,0);
      //max7221 LEDide nullimiseks
      lc.clearDisplay(0);
      attachInterrupt(0, get_u, RISING);
      u = 0;
      rpm = 0;
      timeold = 0;

    }

    void loop() {

      display.setBrightness(7);//7seg display heledus
      display.showNumberDec(rpm);// näitab pöörete numbrit 7deg
displayl
```

```
      i=rpm;
      //1. moodul
      if(i>6000)lc.setRow(0,2,B11111111); else if(i<6000&&i>5750)
lc.setRow(0,2,B11111110);


      if(i>5750&&i<6000)lc.setRow(0,2,B11111110); else
if(i<5750&&i>5500) lc.setRow(0,2,B11111100);
      if(i>5500&&i<5750)lc.setRow(0,2,B11111100); else
if(i<5500&&i>5250) lc.setRow(0,2,B11111000);
      if(i>5250&&i<5500)lc.setRow(0,2,B11111000); else
if(i<5250&&i>5000) lc.setRow(0,2,B11110000);
      if(i>5000&&i<5250)lc.setRow(0,2,B11110000); else
if(i<5000&&i>4750) lc.setRow(0,2,B11100000);


      if(i>4750&&i<5000)lc.setRow(0,2,B11100000); else
if(i<4750&&i>4500) lc.setRow(0,2,B11000000);
      if(i>4500&&i<4750)lc.setRow(0,2,B11000000); else
if(i<4500&&i>4250) lc.setRow(0,2,B10000000);
      if(i>4250&&i<4500)lc.setRow(0,2,B10000000); else
if(i<4250&&i>4000) lc.setRow(0,2,B00000000);
      if(i>4000&&i<4250)lc.setRow(0,1,B11111111); else
if(i<4000&&i>3750) lc.setRow(0,1,B11111110);


      //2. moodul
      if(i>3750&&i<4000)lc.setRow(0,1,B11111110); else
if(i<3750&&i>3500) lc.setRow(0,1,B11111100);
      if(i>3500&&i<3750)lc.setRow(0,1,B11111100); else
if(i<3500&&i>3250) lc.setRow(0,1,B11111000);
      if(i>3250&&i<3500)lc.setRow(0,1,B11111000); else
if(i<3250&&i>3000) lc.setRow(0,1,B11110000);
      if(i>3000&&i<3250)lc.setRow(0,1,B11110000); else
if(i<3000&&i>2750) lc.setRow(0,1,B11100000);


      if(i>2750&&i<3000)lc.setRow(0,1,B11100000); else
if(i<2750&&i>2500) lc.setRow(0,1,B11000000);
      if(i>2500&&i<2750)lc.setRow(0,1,B11000000); else
if(i<2500&&i>2250) lc.setRow(0,1,B10000000);
      if(i>2250&&i<2500)lc.setRow(0,1,B10000000); else
if(i<2250&&i>2000) lc.setRow(0,1,B00000000);
      if(i>2000&&i<2250)lc.setRow(0,0,B11111111); else
if(i<2000&&i>1750) lc.setRow(0,0,B11111110);
      //3. moodul
      if(i>1750&&i<2000)lc.setRow(0,0,B11111110); else
if(i<1750&&i>1500) lc.setRow(0,0,B11111100);
      if(i>1500&&i<1750)lc.setRow(0,0,B11111100); else
if(i<1500&&i>1250) lc.setRow(0,0,B11111000);
      if(i>1250&&i<1500)lc.setRow(0,0,B11111000); else
if(i<1250&&i>1000) lc.setRow(0,0,B11110000);
```

```
      if(i>1000&&i<1250)lc.setRow(0,0,B11110000); else
if(i<1000&&i>750) lc.setRow(0,0,B11100000);
      if(i>750&&i<1000)lc.setRow(0,0,B11100000); else if(i<750&&i>500)
lc.setRow(0,0,B11000000);
      if(i>500&&i<250)lc.setRow(0,0,B11000000); else if(i<500&&i>250)
lc.setRow(0,0,B10000000);
      if(i>250&&i<0)lc.setRow(0,0,B10000000); else if(i<250)
lc.setRow(0,0,B00000000);


      if(i>2000)lc.setRow(0,0,B11111111); else if(i<250)
lc.setRow(0,0,B00000000);
      if(i>4000)lc.setRow(0,1,B11111111); else if(i<2250)
lc.setRow(0,1,B00000000);
      if(i>6000)lc.setRow(0,2,B11111111); else if(i<4250)
lc.setRow(0,2,B00000000);
      //vajalik olukordades, kus pöörete arv hüppab ühelt väärtuselt
teisele nii kiiresti, et jätab ühe terve 8 LED mooduli vahele ja selle
tõttu jäävad ledid kas põlema või kustu


      if (u >= 10) {
     //uuendab pöörete arvu displayl iga 10 signaali tagant
     noInterrupts();//interruptid välja arvutuse ajaks
     rpm = (u*60000/(millis() - timeold))/2;//60000 millis=1min
     timeold = millis();
     interrupts();//interruptid tagasi
     u = 0;
     timeout = 0;


      }else{
      timeout++;
      }
      if (timeout>50) rpm = 0, timeout=0;
    }//vajalik pöörete nullimiseks juhul kui mootor seiskub


    void get_u(){ //interrupti tsükkel jookseb kui tuleb signaal
arduino pinile 2
      u++;   //iga mootori pöördega tekkib 2 interrupti
    }
```