Opinnäytetyö (AMK)

Kone- ja tuotantotekniikka

2021

Lauri Kulmala

AGGREGAATIN MEKAANISEN KAASULÄPÄN MUUNNOS SÄHKÖTOIMISEKSI

TURKU AMK
TURKU UNIVERSITY OF APPLIED SCIENCES

OPINNÄYTETYÖ (AMK) | TIIVISTELMÄ

TURUN AMMATTIKORKEAKOULU

Kone- ja tuotantotekniikka

2021 | 26 sivua, 5 liitesivua

Lauri Kulmala

AGGRGEGAATIN MEKAANISEN KAASULÄPÄN MUUNNOS SÄHKÖTOIMISEKSI

Tämän opinnäytetyön tarkoituksena oli suunnitella ja toteuttaa MOSA GE 7500 -aggregaattiin sähköisesti ohjattu kaasuläppä ja poistaa käytöstä mekaanisesti keskipakovoimalla toimiva säädin. Työn tilaajana toimi Turun ammattikorkeakoulun moottorilaboratorio.

Työ aloitettiin kartoittamalla työssä tarvittavia komponentteja. Kierrosluvun mittaamisessa päädyttiin käyttämään Hall-anturia, ja kaasuläpän ohjaamiseen servomoottoria. Mikrokontrolleriksi valikoitui Arduino-pohjainen ratkaisu, johon on saatavilla runsaasti erilaisia antureita, moottoreita ja muita toimilaitteita. Arduino mikrokontrolleriyksikön ohjelmointi onnistui C- koodilla. Työ suoritettiin Turun ammattikorkeakoulun moottorilaboratoriossa. Servomoottori ja Hall-anturi kiinnitettiin magneeton suojakuoreen. Servomoottorin varteen kiinnitettiin kaasuvivusto. Arduino-mikrokontrolleriin ohjelmoitiin koodi, joka kerää tietoa pyörintänopeudesta ja ohjaa tämän tiedon avulla servomoottoria. Servomoottori ohjaa kaasuläppää ja ylläpitää oikean kierrosnopeuden kuormasta riippumatta.

Opinnäytetyön toteutuksessa oli alkuun haasteita, tavoitteet kuitenkin saavutettiin osittain ja lopputuloksena vanha mekaaninen säädin saatiin korvattua sähkötoimisella säätimellä.

ASIASANAT:

Aggregaatit, Kaasuläppä, Arduino.

BACHELOR´S THESIS | ABSTRACT

TURKU UNIVERSITY OF APPLIED SCIENCES

Mechanical and Producion Engineering

2021 | 26 pages, 5 pages in appendices

Lauri Kulmala

CONVERSION OF THE AGGREGATE'S MECHANICAL THROTTLE FOR ELECTRICAL OPERATION

The purpose of this thesis was to design and implement an electrically controlled throttle for the MOSA GE 7500 unit and to deactivate a mechanically operating centrifugal controller. The work was commissioned by the Turku University of Applied Sciences engine laboratory.

The work was started by mapping the components needed in the work. In measuring the speed, it was decided to use a hall sensor and a servomotor to control the throttle. An Arduino-based solution was chosen as the microcontroller, which there are a wide variety of sensors, motors and other actuators available. The Arduino microcontroller unit was successfully programmed with C code. The work was performed in the engine laboratory of Turku University of Applied Sciences. The servomotor and hall sensor were attached to a magneton safety cover. A throttle linkage was attached to the servomotor arm. The Arduino microcontroller was programmed with a code that collects speed information and uses this information to control the servomotor. The servomotor controls the throttle and maintains the correct speed regardless of the load.

Implementing the thesis there were challenges at first. However, the goals were partially achieved and as a result, the old mechanical controller was replaced with a functioning electric controller.

KEYWORDS:

Aggregate, Throttle body, Arduino

SISÄLTÖ

LIITTEET

KUVAT

KÄYTETYT LYHENTEET

Lyhenne	Lyhenteen selitys
EEPROM	Haihtumaton puolijohdemuisti
EYKK	Ennen yläkuolokohtaa
IR	Infrapuna
I2C	Kommunikaatioprotokolla
KiB	Tavu
OHV	Yläpuolinen venttiili
PWM	Pulssinleveysmodulaatio
Rpm	Kierrosta minuutissa
SRAM	Staattinen RAM-muisti

1 JOHDANTO

Tämän opinnäytetyön tarkoituksena oli suunnitella ja toteuttaa bensiinikäyttöisen aggregaatin kaasuläpän muunnos mekaanisesta sähkötoimiseksi Turun ammattikorkeakoulun opiskelijoiden käyttöön. Työ tehtiin tilaustyönä Turun ammattikorkeakoululle.

Aggregaatti on sähköä tuottava kone, joka toimii esimerkiksi bensiinillä tai dieselöljyllä. Tyypillisiä käyttökohteita ovat alueet tai paikat, joihin ei tule verkkosähköä, kuten kesämökit, työmaat ja maatilat, joissa tarvitaan varavirtaa nopeasti ja helposti. Aggregaatin tarkoitus on pitää tasainen pyörintänopeus, joka takaa oikeataajuisen sähkön tuoton. Tässä opinnäytetyössä käytössä olevan aggregaatin, Mosa GE 7500, pyörintänopeus on 3000 rpm. Muutostyön keskeisin kohta oli pitää tämä pyörintänopeus kuormasta riippumatta sähköisen kaasuläpän ohjaamana.

2 MOSA- AGGREGAATTI

Mosa on italialainen sähkö- ja hitsausgeneraattoreihin erikoistunut valmistaja. Mosa on toimittanut markkinoille luotettavia laitteita jo yli 50 vuoden ajan, vuodesta 1963 lähtien. (Kivirock ammattilehti 2019, 55.)

2.1 Honda GX390 -moottori

Honda GX390 -moottori on 389 cm3 :n suuruinen, yksisylinterinen nelitahtimoottori. Moottori on toteutettu roiskeöljyvoitelulla ja automaattisella puolipuristimella. Moottoriin on saatavana sähkö- tai vetokäynnistys. Venttiilikoneisto on OHV (overhead valve), jossa nokka-akseli sijaitsee kampikoneistossa ja venttiilit saavat toimintavoimansa työntötankojen välityksellä. (Honda Engines 2017.)

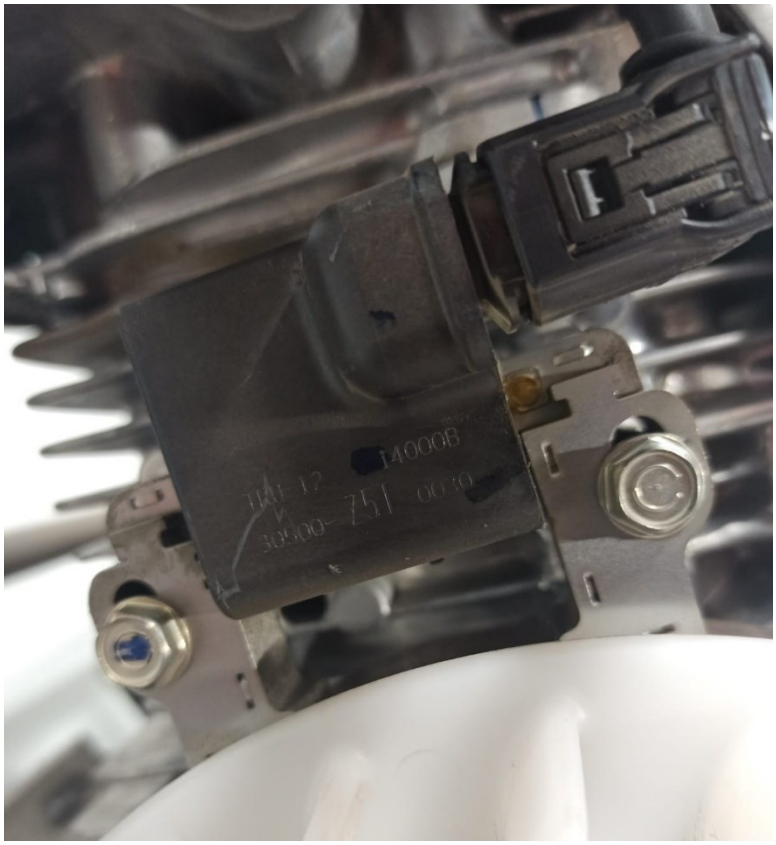
2.2 Kaasutin

Kaasuttimen toiminta perustuu Venturi-ilmiöön. Kaasuttimessa ilmavirta supistetaan kapeampaan putkeen, jolloin ilmavirtaus on nopeampi ja paine pienempi. Ulkoilman paine on suurempi kuin imukanavan, polttoaine kulkeutuu imukanavan kautta sylinteriin. (Wolfram 2011.) Alkuperäinen Keihin BE-mallin kaasutin on läppäohjattu ja sisältää tämän lisäksi tyhjäkäyntipiirin ja kylmäkäynnistysrikastuksen. Kaasuttimen läppää ohjaa säädin. Säädin on linkitetty kaasuläppään tangoilla, jotka ovat yhteyedessä kampikoneistossa olevaan keskipakovoimalla toimivaan tappiin. Tappi työntyy ulospäin kierrosten noustessa ja täten rajoittaa polttoaineseoksen saantia, jolloin kierrokset rajoittuvat 3000 rpm (kuva 1).

Kuva 1. Kaasuttimen vivusto.

2.3 Sytytysjärjestelmä

Moottorissa on yksinkertainen magneettosytytys, eikä rakenne tarvitse ulkopuolista virtaa toimiakseen. Sen heikkoutena on kuitenkin säätömahdollisuuksien puuttuminen, sytytysteho on heikko eikä sytytysennakon säätömahdollisuutta ole. Sytytysennakko on kiinteä 10° EYKK. Puolassa on sisäänrakennettu kierrostenrajoitin, joka rajoittaa maksimi kierrosluvun 4500 rpm (kuva 2). (Stud racing.)



Kuva 2. Sytytyspuola.

Kierrostenrajoitin on tärkeä, sillä kun alkuperäinen säädin poistetaan käytöstä ja kaasuläppää aloitetaan ohjaamaan sähköisesti, niin rajoitin suojaa moottoria ylikierroksilta, jotka voisivat vaurioittaa kampi- tai venttiilikoneistoa (kuva 3).

Kuva 3. Säädin ja ohjaustangot.

3 ARDUINO JA KOMPONENTTIEN VALINTA

Arduino on mikro-ohjain/elektroniikka-alusta jota ohjelmoidaan c ja c++:aan perustuvalla Arduino-ohjelmointikielellä. Laitteisto käyttää 8-bittistä Atmel AVR mikro-ohjainta, jonka pinneihin voi kytkeä erilaisia antureita, moottoreita ja muita komponentteja. Arduino-laitteita on monia eri malleja, jotka eroavat toisistaan. Eroja ovat EEPROMin, keskusmuistin ja Flash-muistin sekä digitaalisten ja analogisten pinnien määrissä. (Arduino 2018.). Tähän opinnäytetyöhön valittiin Arduino Uno -mallinen mikro-ohjain, jossa on Atmega328 suoritin, 32 KiB flash muistia, EEPROmia 1 KiB ja SRAMia 2 KiB. Lisäksi digitaalisia sisääntuloja on 14kpl ja analogisia 6kpl. Harkinnassa oli myös tehdä työ Raspberry Pi:lla, mutta Arduino oli selkeästi järkevämpi vaihtoehto yksinkertaisuuden vuoksi. Kaikki tarvittavat komponentit pystyttiin kytkemään suoraan Arduinoon ilman lisäkortteja. Näin työhön ei tarvinnut kokonaista pientä tietokonetta. Lisäksi Arduinossa on analogisia sisääntuloja, joita monessa Raspberryssä ei ole.

3.1 Pyörintänopeuden mittaus

Kampiakselin pyörintänopeuden mittaamiseen käytettiin Hall-tyyppistä anturia. Hall-anturin toiminta perustuu Hall-ilmiöön. Anturin luomaa magneettikenttää häiritään toisella magneetilla, tässä tapauksessa vauhtipyörässä olevalla magneetilla, Hall-anturi antaa suorakaiteen muotoisen signaalipulssin aina moottorin pyörähtäessä kerran ympäri. (ElectronicsTutorials 2019.) Tietoa tarvitaan, jotta moottori saadaan pitämään tasainen 3000 rpm käyntinopeus kun ohjaamme kaasuläppää sähköisesti servomoottorilla. Tähän työhön valittiin SS443A, joka on Honeywellin valmistama digitaalinen Hall-anturi. Anturi tarvitsee myös vastuksen, jolla vahvistetaan anturin antamaa signaalia. Vaihtoehtona oli myös IR-sensorin käyttäminen, jolloin vauhtipyörään tai johonkin pyörivään osaan oltaisiin tarvittu valko-musta-valkotaustainen osio, josta IR-sensori olisi lukenut pyörintänopeutta. Hall-anturiin päädyttiin, koska vauhtipyörä sisälsi jo valmiiksi magneetin, joten toteutus oli huomattavasti helpompaa koska tarvittiin vain anturi (kuva 4).

Kuva 4. Hall-anturi (Elfa distrelec 2020a).

3.2 Kaasuläpän ohjaus

Kaasuläppä säätää moottorille menevää ilman määrää. Kun kaasuläppä on täysin auki, polttoaine-ilmaseosta virtaa moottoriin maksimimäärä, mikä mahdollistaa moottorin maksimikuormituksen (The Editors of Encyclopaedia Britannica 1998). Läppää ohjattiin alkuperäisessä aplikaatiossa mekaanisesti ohjaustangolla, joka oli yhteydessä säätimeen. Säädinjärjestelmän tilalle asennettiin pienikokoinen Seed Studion servomoottori, joka toimii 4.8-6 v jännitteellä. Servomoottorin varsi on kytketty kaasuläppää käyttävään tankoon, jolloin servoa kääntämällä kaasuläppä kääntyy tangon välityksellä. Servomoottori on kytketty Arduinoon, jolla ohjataan servomoottorin kääntyvyyttä sekä toimintaa. Vaihtoehtona oli myös käyttää askelmoottoria, jonka etuna olisi ollut säädön tarkkuus verrattuna servomoottorin. Askelmoottorin käyttö olisi kuitenkin vaatinut lisäkortin hankkimista sekä monimutkaisempaa ohjelmointia. Servomoottori todettiin riittävän tarkaksi ja säätöjen sekä kulmien ohjelmointi oli sillä helpompaa, koska servomoottorin voi kytkeä Arduinoon suoraan ilman lisäkortteja (kuva 5).

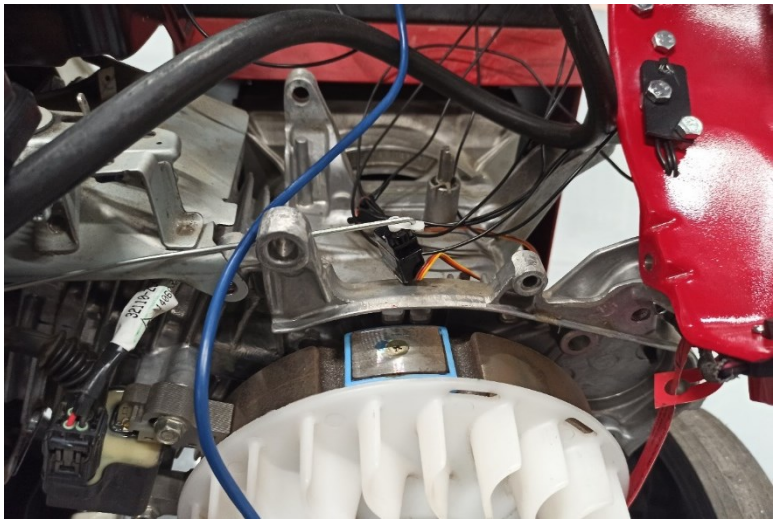
Kuva 5. Servomoottori (Elfa distrelec 2020b).

3.3 Näyttö

Moottorin pyörimisnopeuden näytöksi valittiin Adafruit 128x64 OLED-näyttö. Näyttö liitettiin mikrokontrolleriin I2C protokollaa käyttäen. Tämä tapa valittiin, koska vaihtoehtoinen SPI -väylä olisi vaatinut useampien johtimien käyttöä mikrokontrollerin ja näytön väillä, lisäksi I2C protokollan käyttö tässä sovelluksessa on riittävän nopea.

4 KOMPONENTTIEN ASENNUS MOOTTORIIN

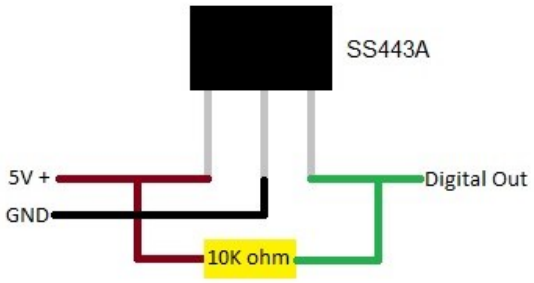
Asennus oli hieman haasteellista, koska käytössä ei ollut minkäänlaista asennussarjaa, vaan kaikki muutokset olivat uniikkeja ja kiinnikkeet täytyi suunnitella itse, kuten myös paikat mihin komponentit tulevat kiinni. Hall-anturi ja servo asennettiin magneeton suojakuoreen. Servomoottorin asennuksessa käytettiin apuna kahta L-muotoista levyä, joiden väliin servopuristus kiinnitettiin. Hall-anturi kiinnitettiin magneeton suojakuoren sisäpuolelle, samoilla pulteilla millä L-levyt ovat kiinni kuoren ulkopuolella. Alla on kuva moottorista ja asennuksista magneeton suojakuoressa (kuva 6). Arduino on asennettuna omaan elektroniikka-asennuslaatikkoon, joka on moottorin hoitotasolla.



Kuva 6. Asennuksien suunnittelu.

4.1 Pyörimisnopeustunnistimen asennus

Moottorin magneeton suojakuoreen porattiin kaksi kappaletta kuuden millimetrin reikää ja lisäksi yksi reikä tarvittaville johdoille. Hall-anturi asennettiin elektroniikkalevyyn poraamalla reiät Hall-anturin jaloille. Anturiin tinattiin lisäksi signaalin vahvistamiseen tarkoitettu 10k ohm vastus. Alla olevassa kuvassa (kuva 7) Hall-anturin kytkentä: digitaalinen ulostulo menee Arduino-mikrokontrollerin digitaalisen sisääntulon toiseen pinniin.



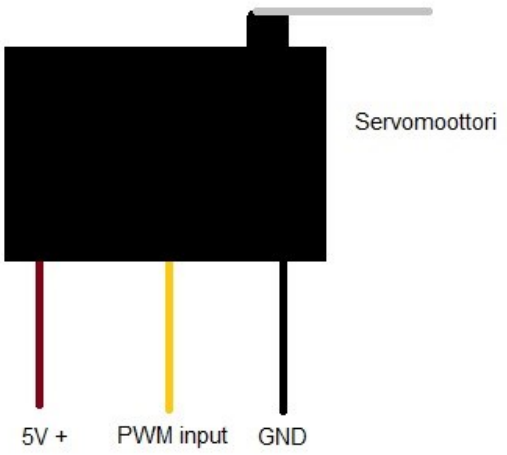
Kuva 7. Hall-anturin kytkentä.

Hall-anturin asennus suojakuoren sisällä (kuva 8).

Kuva 8. Asennus suojakuoren sisäpuolella.

4.2 Kaasuläppää ohjaavan servomoottorin asennus

Servomoottori asennettiin magneeton suojakuoreen L-raudoilla. Suojakuoreen porattiin 4 kpl 6 mm reikää, johon raudat kiinnitettiin pulteilla ja muttereilla. Servomoottorin varteen porattiin 2 mm reikä, johon kiinnitettiin kaasuttimeen menevä tanko joka ohjaa kaasuläpän asentoa. Servomoottoriin kytkentä (kuva 9). PWM input kytkettiin Arduinon digitaalisen ulostulon yhdeksänteen pinniin.



Kuva 9. Servomoottorin kytkentä.

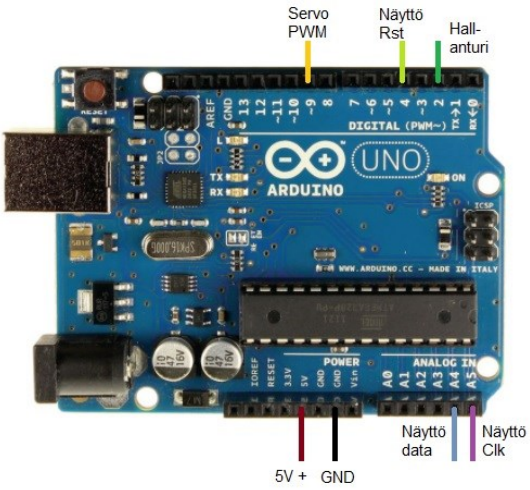
4.3 Näytön asennus

Elektroniikka-asennuksiin tarkoitettuun laatikkoon porattiin reikä näytölle, johon näyttö ruuvattiin kiinni. Johdot kolvattiin suoraan näytön piirilevyyn kiinni, näytöstä kolvattiin pinnit SJ1 ja SJ2 yhteen, jotta näyttö saatiin käyttämään I2C protokollaa. I2C protokollassa käytetään näytön kahdeksasta pinnistä viittä (kuva 10).

Kuva 10. Oled-näyttö (Elfa distrelec 2020c).

4.4 Arduino asennus ja kytkennät

Arduino-mikrokontrolleri asennettiin elektroniikka-asennuslaatikkoon. Laatikkoon porattiin sopivat reiät johdoille sekä kiinnityksille. Arduino-mikrokontrolleriin kytkettiin kaikki komponenteista tulevat johdot kuvan 11 mukaisesti. (kuva 11).



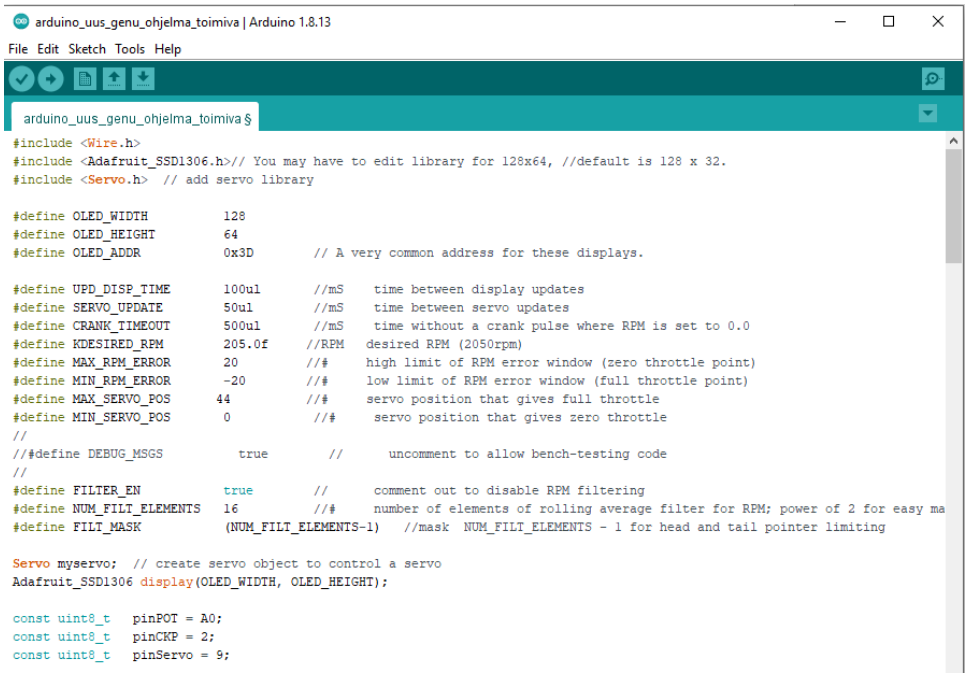
Kuva 11. Mikro-ohjainkortti kytkennät (Elfa distrelec 2020d).

5 ARDUINO-MIKROKONTROLLERIN OHJELMOINTI JA SÄÄTÖ

Arduinon ohjelmointi tehtiin Arduino IDE-ohjelmalla. Arduino liitettiin tietokoneeseen USB-pistokkeella, josta data vietiin mikrokontrolleriin. Arduino saa käytettävän virran USB-portin kautta, joten ulkoista virtalähdettä ei välttämättä tarvita. Arduinosta löytyy myös erillinen virtakaapelin paikka. Arduinoa ohjelmoidaan Open Source -pohjaisessa viitekehyksessä, joka muistuttaa läheisesti C-kieltä. Tämä alusta tukee laajaa joukkoa erilaisia mikroprosessoreita ja kehitysalustoja.

5.1 Käytettävän ohjelman asetukset!

Ohjelma pyrittiin tekemään mahdollisimman yksinkertaiseksi, jolloin vältyttäisiin mahdollisilta ohjelmointivirheiltä. Ensiksi määriteltiin raja-arvot kaikille komennoille ja liikkeille mitä komponentit tekevät (kuva 12).



Kuva 12. Määritelmät.

Määritelmien jälkeen tehtiin ohjelma, johon koodattiin käytettävät loopit, joita ohjelma käy lävitse kokoajan ja arvojen muuttuessa tekee määritellyn toiminnon. Servomoottorin

maksimaalinen kääntyvyys tässä asennuksessa oli 55 astetta, jolloin kaasuläppä oli täysin auki ja vastaavasti 0 astetta, jolloin kaasuläppä oli täysin kiinni-asennossa. Ohjelmaan myös määriteltiin päivitysajat näytölle ja servomoottorille sekä suodatus kierrosluvulle.

5.2 Pyörintänopeuden säädin

Pyörintänopeuden säätimen toimintaa pystytään kontrolloimaan Hall-anturilla, josta saadaan moottorin pyörintänopeus kampiakselilta. Tätä pyörintänopeustietoa voidaan hyödyntää siten, että kierrosluvun laskiessa alle asetetun korjausarvon, mikrokontrolleri laskee että kierroksia on joko lisättävä tai vähennettävä, ja tällöin ohjaa servomoottorin joko avaamaan/sulkemaan kaasuläppää niin kauan, että kierrokset ovat taas raja-arvojen välissä. Tämä kaikki tapahtuu hyvin nopeasti eikä moottori pääse sammumaan, vaikka kuormaa lisättäisiin hyvinkin nopeasti. Pyörintänopeuden säätimen koodi (kuva 13).

```
void UpdateServo( void )
{
    static uint32_t
        timeServo = 0;
    uint32_t
        timeNow = millis();

    if( (timeNow - timeServo) >= SERVO_UPDATE )
    {
        timeServo = timeNow;

        int RPMError = (int)(ffiltRPM - KDESIRED_RPM);
        int cRPMError = constrain( RPMError, MIN_RPM_ERROR, MAX_RPM_ERROR );

        //error is constrained to a min of -200RPM and a max of +200RPM
        //a negative error means the actual RPM is LESS THAN than the desired RPM (we want the servo to open the throttle)
        //a positive error means the actual RPM is MORE THAN than the desired RPM (we want the servo to close the throttle)
        //
        int servoPos = map( cRPMError, MIN_RPM_ERROR, MAX_RPM_ERROR, MAX_SERVO_POS, MIN_SERVO_POS );
        myservo.write( servoPos );

    }//if

}//UpdateServo
```

Kuva 13. Pyörintänopeuden säätimen koodi.

6 KOEKÄYTTÖ JA HAVAINNOT

Ensimmäisen koekäytön aikana moottori ei pystynyt ylläpitämään kierroslukua, vaan kierrokset laskivat huomattavasti. Koodin raja-arvoihin tehtiin korjauksia, jonka jälkeen moottori saatiin ylläpitämään haluttua tasaista kierroslukunopeutta ja kierrokset pysyivät stabiilina kuormasta riippumatta. Moottoria kuormitettiin koeajossa säädettävällä lämpöpuhaltimella, jonka tehontarve oli suurimmillaan 5 kW.

Koekäytössä testausolosuhteet olisivat voineet olla paremmat, sillä moottoria jouduttiin testaamaan ulkona moottorilaboratorion viimeistelytöiden vuoksi. Arduino -kontrolleriin vaatima sähkönsyöttö kytekttiin koeajossa kannettavaan tietokoneeseen, mutta ulkoisen virtalähteen käyttö on mahdollista. Virranottoa aggregaatista myös suunniteltiin, mutta aggregaatin tuottaman sähkön laatu arveltiin heikoksi, eikä tätä kokeiltu.

7 YHTEENVETO

Opinnäytetyön tavoite saavutettiin osittain ja moottorin alkuperäinen säädin pystyttiin korvaamaan servomoottori ohjatulla säätimellä. Pyörintänopeuden mittaukseen ja näyttöä ei saatu toimimaan eikä pyörintänopeuden arvoa pysty tallentamaan ulkoisella laitteella. Projektiin valittu Arduino mikrokontrolleri soveltui vaihtoehdoksi tämän kaltaiseen työhön, sillä laitteistoon on saatavilla paljon erilaisia komponentteja edulliseen hintaan.

Ohjelmaa tulisi vielä jatkossa kehittää, sillä vaikka moottori saatiin pitämään tasainen kierrosluku niin Arduinoin ohjaamalla näytöllä oleva lukema ei vastannut todellista kierroslukua, vaan se oli 200 rpm Vastaavasti oskilloskoopilla tarkastettuna puolan ja tulpan välisestä johdosta lukema todelliset kierrokset olivat noin 3300-3600 rpm. Ongelmaa yritettiin ratkaista poistamalla käytöstä pyörintänopeusanturin signaalin suodatin ohjelmasta, tämä toimenpide ei kuitenkaan ratkaissut ongelmaa.

Aggregaatin mekaaninen säädin saatiin korvattua sähkötoimisella säätimellä joka toimii odotetulla tavalla ja sen ansiosta säädintä on mahdollista ohjelmoida tietokoneella. Jatkossa pyörintänopeuden mittaus pitää korjata, sekä laitteistoa kehittää niin, että pyörintänopeuden loggaus ulkoiseen tiedonkeruulaitteiseen onnistuu ja pyörintänopeuden asetusarvo voidaan asettaa ilman tietokonetta.

LÄHTEET

Arduino 2018. Koodaus. Viitattu 2.12.2020 https://www.circuito.io/blog/arduino-code/

Arduinomanuals 2020. Arduino. Viitattu 2.12.2020 https://www.arduino.cc/en/Guide/ArduinoUno

Electronicstutorials 2019. Hall Effect Sensor. Viitattu 2.12.2020 https://www.electronics-tutorials.ws/electromagnetism/hall-effect.html

Elfa distrelec 2020a. Hall-anturi. Viitattu 2.12.2020 https://www.elfadistrelec.fi/fi/hall-anturi-30v-20ma-digitaalinen-virranotto-honeywell- ss443a/p/17349087?q=&pos=4&origPos=4&origPageSize=10&track=true

Elfa distrelec 2020b. Grove-servo. Viitattu 2.12.2020 https://www.elfadistrelec.fi/fi/grove-servo-seeed-studio-316010005/p/30069966?queryFromSuggest=true

Elfa distrelec 2020c. OLED-näyttö. Viitattu 2.12.2020. https://www.elfadistrelec.fi/fi/mustavalkoinen-128x64-oled-naeyttoe-adafruit-938/p/30091227?queryFromSuggest=true

Elfa distrelec 2020d. Mikro-ohjainkortti kytkennät. Viitattu 2.12.2020 https://www.elfadistrelec.fi/fi/mikro-ohjainkortti-uno-arduino-a000066/p/11038919?queryFromSuggest=true

Grove-servo, Seeed Studio 2015. Servomoottori. Viitattu 1.12.2020 https://www.elfadistrelec.fi/Web/Downloads/_t/ds/316010005_eng_tds.pdf

Honda 2020. Honda GX390. Viitattu 2.12.2020 https://www.honda-engines-eu.com/c/portal/layout?p_l_id=11194&p_v_l_s_g_id=0&e=19

I2C protokolla 2020. Näyttö. Viitattu 2.12.2020 https://www.digikey.fi/fi/articles/why-the-inter-integrated-circuit-bus-makes-connecting-ics-so-easy

Kivirock maarakennus- ja kaivosalan ammattilehti 1/2019, 55. Viitattu 2.12.2020

Mosa GE 7500 HBSL 2014. Manualslib. Viitattu 2.12.2020 https://www.manualslib.com/products/Mosa-Ge-7500-Hbsl-Avr-5536094.html

SS443A 2020. Hall-anturi. Viitattu 1.12.2020 https://www.elfadistrelec.fi/Web/Downloads/s4/00/mnSS400.pdf

Stud racing. Viitattu 2.12.2020 http://www.studzracing.com/category-s/119.htm

The Editors of Encyclopaedia Britannica 7/1998. Viitattu 2.12.2020 https://www.britannica.com/technology/carburetor

Wolfram 2011. Venturi-ilmiö. Viitattu 26.11.2020 https://demonstrations.wolfram.com/TheVenturiEffect/

Ohjelmakoodi

```
#include <Wire.h>
#include <Adafruit_SSD1306.h>// You may have to edit library for 128x64,
//default is 128 x 32.
#include <Servo.h>  // add servo library

#define OLED_WIDTH          128
#define OLED_HEIGHT         64
#define  OLED_ADDR           0x3D        // A very common address for
these displays.

#define  UPD_DISP_TIME       100ul       //mS    time between display
updates
#define SERVO_UPDATE        50ul        //mS    time between servo up-
dates
#define  CRANK_TIMEOUT       500ul       //mS    time without a crank
pulse where RPM is set to 0.0
#define KDESIRED_RPM        205.0f     //RPM   desired RPM (2050rpm)
#define MAX_RPM_ERROR       20         //#     high limit of RPM error
window (zero throttle point)
#define  MIN_RPM_ERROR       -20        //#     low limit of RPM error
window (full throttle point)
#define MAX_SERVO_POS      55          //#     servo position that gives
full throttle
#define  MIN_SERVO_POS       0           //#     servo position that
gives zero throttle
//
//#define  DEBUG_MSGS          true        //      uncomment to allow
bench-testing code
//
#define FILTER_EN           false       //      comment out to disable
RPM filtering
#define  NUM_FILT_ELEMENTS   16          //#     number of elements of
rolling average filter for RPM; power of 2 for easy masking
#define             FILT_MASK                       (NUM_FILT_ELEMENTS-
1)   //mask  NUM_FILT_ELEMENTS - 1 for head and tail pointer limiting

Servo myservo;  // create servo object to control a servo
Adafruit_SSD1306 display(OLED_WIDTH, OLED_HEIGHT);

const uint8_t   pinPOT = A0;
const uint8_t   pinCKP = 2;
const uint8_t   pinServo = 9;

HardwareSerial *serialConsole = (HardwareSerial *)&Serial;

int val;    // variable to read the value from the analog pin

float
    fRPM,
    ffiltRPM,
    fRPMSum,
    grfRPMHistory[NUM_FILT_ELEMENTS];
uint8_t
    headPtr,
    tailPtr;
```

```
volatile bool
    bWaitFirst = true,
    bPulse = false;
volatile uint32_t
    revPeriod;

char
    szStr[80];

void isr( void )          //interrupt service routine
{
    static uint32_t
        timeLastPulse;

    if( bWaitFirst )
    {
        timeLastPulse = micros();
        bWaitFirst = false;

    }//if
    else
    {
        uint32_t timeNow = micros();
        revPeriod = micros() - timeLastPulse;
        timeLastPulse = timeNow;

        bPulse = true;

    }//else

}//isr

void setup( void )
{
    serialConsole->begin( 115200 );

    display.begin( SSD1306_SWITCHCAPVCC, OLED_ADDR );
    display.clearDisplay();

    pinMode( pinCKP, INPUT_PULLUP );
    attachInterrupt(    digitalPinToInterrupt(pinCKP),    isr,    RISING
);  //attaching the interrupt
    myservo.attach( pinServo );                                          //
attaches the servo on pin 9 to the servo object
    myservo.write(                                              MAX_SERVO_POS
);                                  //full-throttle for engine start

    for( uint8_t i=0; i<NUM_FILT_ELEMENTS; i++ )
        grfRPMHistory[i] = 0.0;
    fRPMSum = 0.0;
    headPtr = NUM_FILT_ELEMENTS - 1;
    tailPtr = 0;

}//setup

void loop( void )
{
    //val = analogRead( potpin );             // reads the value of the
potentiometer (value between 0 and 1023)
```

```
    //val = map( val, 0, 1023, 0, 10 );       // scale it to use it with
the servo (value between 0 and 180)

    CalcRPM();
    UpdateServo();
    UpdateDisplay();

}//loop

void CalcRPM( void )
{
    static uint32_t
        timeOut;
    uint32_t
        timeNow,
        _period;

    timeNow = millis();

    if( bPulse )
    {
        noInterrupts();
        bPulse = false;
        _period = revPeriod;
        interrupts();

        fRPM = 60.0f / ((float)_period * 1.0E-06);

        timeOut = timeNow;

    }//if
    else if( (timeNow - timeOut) >= CRANK_TIMEOUT )
    {
        noInterrupts();
        bWaitFirst = true;
        interrupts();
        timeOut = timeNow;

#ifndef DEBUG_MSGS
        fRPM = 0.0;
#endif

    }//if

    ffiltRPM = FilterRPM( fRPM );

}//CalcRPM

void UpdateServo( void )
{
    static uint32_t
        timeServo = 0;
    uint32_t
        timeNow = millis();

    if( (timeNow - timeServo) >= SERVO_UPDATE )
    {
        timeServo = timeNow;
```

```
        int RPMError = (int)(ffiltRPM - KDESIRED_RPM);
        int cRPMError = constrain( RPMError, MIN_RPM_ERROR, MAX_RPM_ER-
ROR );

        //error is constrained to a min of -200RPM and a max of +200RPM
        //a negative error means the actual RPM is LESS THAN than the
desired RPM (we want the servo to open the throttle)
        //a positive error means the actual RPM is MORE THAN than the
desired RPM (we want the servo to close the throttle)
        //
        int servoPos = map( cRPMError, MIN_RPM_ERROR, MAX_RPM_ERROR,
MAX_SERVO_POS, MIN_SERVO_POS );
        myservo.write( servoPos );

    }//if

}//UpdateServo

float FilterRPM( float fCurrRPM )
{
    fRPMSum -= grfRPMHistory[headPtr];
    fRPMSum += fCurrRPM;
    grfRPMHistory[tailPtr] = fCurrRPM;

    headPtr = (headPtr + 1) & FILT_MASK;
    tailPtr = (tailPtr + 1) & FILT_MASK;

#ifdef FILTER_EN
    return( fRPMSum / (float)NUM_FILT_ELEMENTS );
#else
    return fCurrRPM;
#endif

}//FilterRPM

void UpdateDisplay( void )
{
    static uint32_t
        timeDisp;
    uint32_t
        timeNow = millis();

    if( (timeNow - timeDisp) >= UPD_DISP_TIME )
    {
        display.clearDisplay();
        display.setTextSize(2);
        display.setTextColor(WHITE);
        display.setCursor(0, 0);// Vertical, Horizontal.
        display.println("RPM:");
        display.setTextSize(5);
        display.setTextColor(WHITE);
        display.setCursor(0,25);
        uint16_t rpm = (uint16_t)ffiltRPM;
        display.println(rpm);
        display.display();

#ifdef DEBUG_MSGS
        dtostrf( fRPM, 6, 1, szStr );
        serialConsole->print( "RPM: " ); Serial.print( szStr );
```

```
        dtostrf( ffiltRPM, 6, 1, szStr );
        serialConsole->print( "\tfiltRPM: " ); Serial.print( szStr );
        serialConsole->print( "\tServo Pos: " ); Serial.print( servoPos
);
        serialConsole->println();

        fRPM = fRPM + 2.0;
        if( fRPM > 400.0 )
            fRPM = 0.0;
#endif
    }//if

}//UpdateDisp
```