VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES

Lauri Vuori

TAAJUUSMUUTTAJAN MODULAATTORIN TOTEUTUS MIKRO-OHJAIMELLA

Tekniikka
2023

VAASAN AMMATTIKORKEAKOULU
Tietotekniikka

TIIVISTELMÄ

Tekijä	Lauri Vuori
Opinnäytetyön nimi	Taajuusmuuttajan modulaattorin toteutus mikro-ohjaimella
Vuosi	2023
Kieli	Suomi
Sivumäärä	34
Ohjaaja	Jani Ahvonen

Tämä opinnäytetyö on tehty Vaasan ammattikorkeakoulun toimeksiannosta. Tämän opinnäytetyön tavoitteena oli tutkia tapoja toteuttaa taajuusmuuttajan modulaattori, sekä tavan löydyttyä ohjelmoida ja testata kyseinen toteutus. Näin ollen tarkoituksena oli selvittää, onko NUCLEO-L152RE -kontrollerilla mahdollista toteuttaa taajuusmuuttajan modulaattori, jonka avulla voidaan ohjata induktiomoottoria vaihtelevalla nopeudella.

Taajuusmuuttajat ovat merkittävässä osassa teollisuutta. Taajuusmuuttaja ei ole uusi keksintö, mutta yksinkertaisia toteutuksia taajuusmuuttajan modulaattorille ei ole tällä hetkellä juurikaan saatavilla. Toteutetussa työssä tärkeimmät kontrollerin ominaisuudet ovat ajastimet sekä keskeytykset.

Tästä opinnäytetyöstä saadun tiedon avulla on mahdollista toteuttaa yksinkertainen ja halpa taajuusmuuttajan modulaattori käyttäen NUCLEO-L152RE -mikrokontrolleria. Tietoa voidaan hyödyntää opetuksessa ja taajuusmuuttajan ja sähkömoottorin toiminnan demonstroimisessa.

Opinnäytetyössä jatkotutkimuskohteena voisi olla valmiiksi koodattujen arvojen laskeminen ajon aikana ja koko toteutuksen tekeminen nopeammalla kontrollerilla. Tämä osa tulisi tutkia tarkemmin nopeammalla prosessorilla.

Avainsanat mikrokontrolleri, ajastin, keskeytys

VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES
Information technologies

ABSTRACT

Author	Lauri Vuori
Title	Implementation of a Variable Frequency Drive modulator with a Microcontroller.
Year	2023
Language	Finnish
Pages	34
Name of Supervisor	Jani Ahvonen

This thesis has been commissioned by Vaasa University of Applied Sciences. The purpose of the thesis was to investigate ways to implement a variable speed drive modulator and, once found, to program and test the implementation. Therefore, the goal was to determine whether it is possible to implement a variable speed drive modulator using the NUCLEO-L152RE controller, which can control an induction motor at varying speeds.

Variable speed drives are in a significant role in the industry. While variable speed drives are not a new invention, there are currently few simple implementations of variable speed drive modulators available. In the implemented work, the most important controller features are timers and interrupts.

Based on the thesis, it is possible to implement a simple and inexpensive variable speed modulator using the NUCLEO-L152RE microcontroller. This information can be utilized in teaching and demonstrating the operation of a variable speed drive and induction motor.

Further research could include calculating pre-coded values during operation and implementing the entire solution with a faster controller. This aspect should be further investigated with a faster processor.

Keywords Microcontroller, timer, and interrupt

SISÄLLYS

1 JOHDANTO

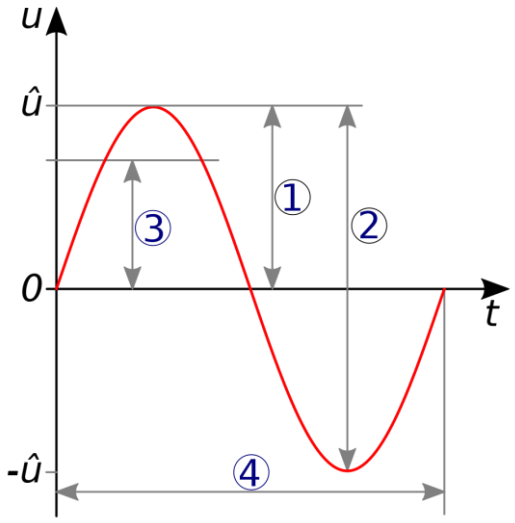
Tämän opinnäytetyön tarkoituksena on selvittää, voidaanko taajuusmuuttajan modulaattori toteuttaa NUCLEO-L152RE -mikrokontrollerilla. Taajuusmuuttajan modulaattori ohjelmoidaan C-kielellä NUCLEO-L152RE -kehitysalustalle ja testataan oskilloskoopilla. Työn pohjana toimii simuloitu toteutus taajuusmuuttajan modulaattorista sekä kytkimistä. Opinnäytetyön ohjelmistokoodissa käytetään mikrokontrollerin ajastimia, joiden avulla pystytään toteuttamaan pulssinleveysmodulaatio. Kontrollerin keskeytyksien yhdistäminen ajastimiin mahdollistaa jatkuvan modulaation pulssisuhteen muuttamisen.

Taajuusmuuttajien käyttö teollisuudessa tarjoaa merkittäviä etuja, kuten korkeampaa energiatehokkuutta, optimoituja prosesseja ja pidempiaikaisia laitteistoja. Edellä mainittujen seikkojen vuoksi taajuusmuuttajat ovat tärkeässä asemassa kestävän teknologian ja optimoidun teollisuuden kannalta. Sähkömoottorin nopeutta säätämällä taajuusmuuttajat voivat vähentää energiankulutusta ja tehostaa prosessien hyötysuhdetta merkittävästi. Kustannusten vähentymisen lisäksi voidaan saada merkittävästi vähemmän ympäristöhaittoja. Paremmalla säädettävyydellä voidaan parantaa tuotteiden laatua ja vähentää hävikkiä. Lukuisien hyötyjen lisäksi myös taajuusmuuttajat vähentävät laitteistojen turhaa kulumista, huoltokustannuksia sekä käyttökatkoksia.

2 VAIHTOVIRTA

Vaihtovirta (AC) on sähkövarauksen virtaus, joka tasaisin väliajoin kääntyy. Se esimerkiksi alkaa nollasta, josta se kasvaa maksimiin, minkä jälkeen se palaa nollaan ja kääntyy vastakkaiseen suuntaan nousten huippuunsa. Se toistaa tätä samaista sykliä loputtomasti. Aikaväliä, joka kuluu tietyn arvon saavuttamisen välillä kahdessa onnistuneessa syklissä, kutsutaan jaksonajaksi eli periodiksi. Periodien lukumäärä per sekunti kutsutaan taajuudeksi ja maksimi arvo suuntaan on amplitudi eli värähdyslaajuus. [1]

Kuviossa 1 on esitetty siniaalto, jossa numerolla 1. on amplitudi, 2. huipusta huippuun amplitudi (peak-to-peak) ja 4. numerolla esitetään jaksonaika.



Kuvio 1. Siniaallon kuuvaja. [2]

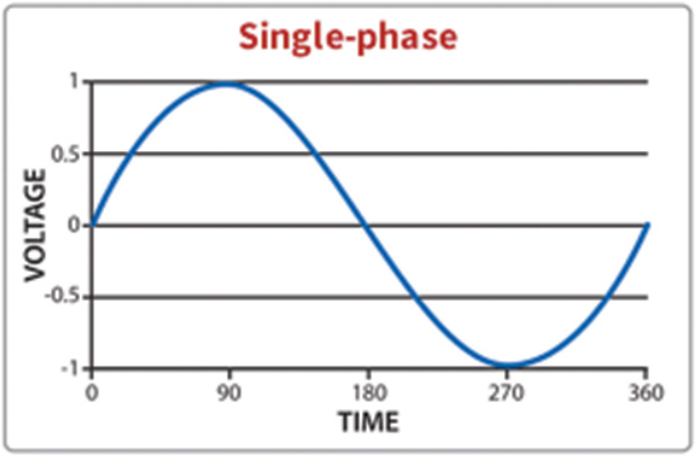
[1] Gregersen, Alternating current, 2023
[2] Wikipedia, Alternating current, 2023

Kaavan 1 avulla voidaan ilmaista sinimuotoisen aallon korkeus y ajan t ja paikan x funktion avulla.

$$y(t) = A * \sin(kx - \omega t - \theta) \quad (1)$$

Yksivaiheinenvirta on yleisimmin käytetty jännitteen muoto ja sitä käytetään pääasiassa kodeissa. Kolmivaiheinen järjestelmä on yleinen teollisuudessa, joissa tarvitaan suuria tehokuormia. Yksivaihejärjestelmät käyttävät vaihtovirtaa (AC), tyypillisesti se on taajuudeltaan 50-60 hertsiä. [3]

Kuviossa 2 on esitetty yksivaihevirta, jossa y-akselilla jännite ja vaaka-akselilla aika.

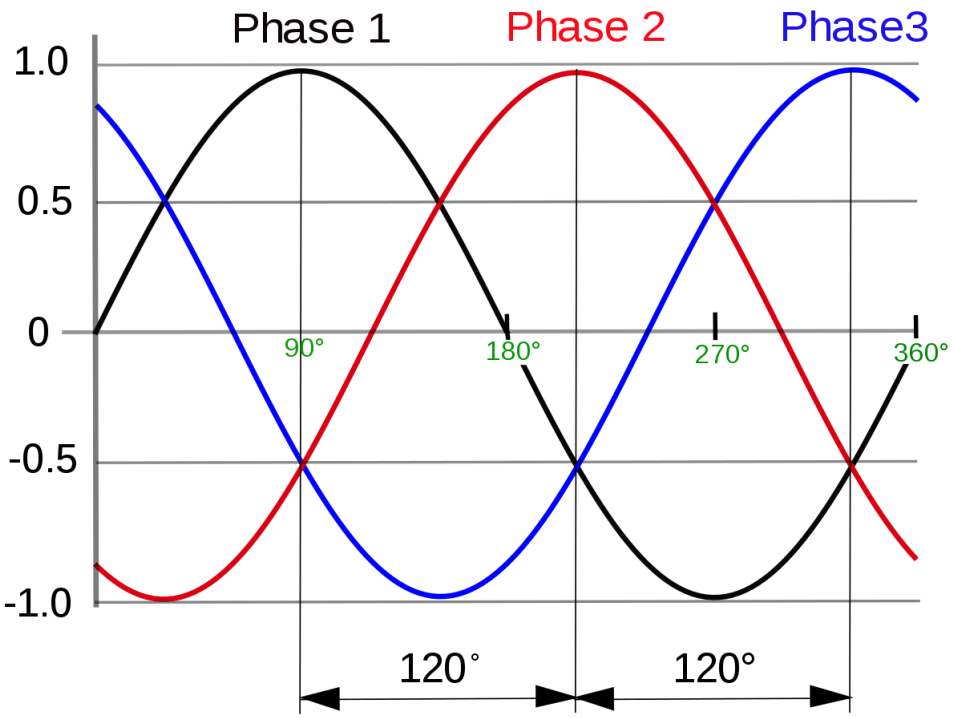


Kuvio 2. Yksivaihevirta kuvaajassa. [4]

[3] National Fire Chiefs Council, The difference between a single-phase and a three-phase power systems, 2023

[4] National Fire Chiefs Council, The difference between a single-phase and a three-phase power systems, 2023

Kolmivaihevirta voidaan yksinkertaistettuna pitää kolmena erillisenä yksivaihevirtana, jossa niiden kärjet ovat 120° erillään toisistaan. Kuviossa 3 on esiteltynä kolme erillistä vaihetta. [5]



Kuvio 3. Kolmivaihevirta kuvaajassa.[6]

[5] National Fire Chiefs Council, The difference between a single-phase and a three-phase power systems, 2023
[6]Wikipedia, Three-phase electric power, 2023

3 TAAJUUSMUUTTAJA

Taajuusmuuttaja (VFD) on moottoriohjain, joka suojaa ja kontrolloi moottorin nopeutta. Sen avulla voidaan hallita moottoria sen käynnistyessä ja pysähtyessä, sekä koko ajon aikana tuottamalla säädettävää taajuutta. Tämän vuoksi ohjainta kutsutaan taajuusmuuttajaksi.[7]

Taajuusmuuttajan hyötyjä ovat nopeuden mitoittaminen prosessin vaatimusten mukaiseksi, energiansäästö ja järjestelmien tehokkuuden parantaminen. Oikean nopeuden saavuttaminen vähentää koneiden mekaanista rasitusta ja lisää niiden käyttöikää.[8]

Tärkein ominaisuus taajuusmuuttajan toiminnassa on pulssinleveysmodulaatio (Pulse-width modulation, PWM), jonka avulla taajuusmuuttajan transistorit tai kytkimet kytketään päälle ja pois useita kertoja. Pulssinleveysmodulaatio muuntelee näin lähtötaajuutta ja -jännitettä ohjaamalla ja muuttamalla pulssien leveyttä. [9]

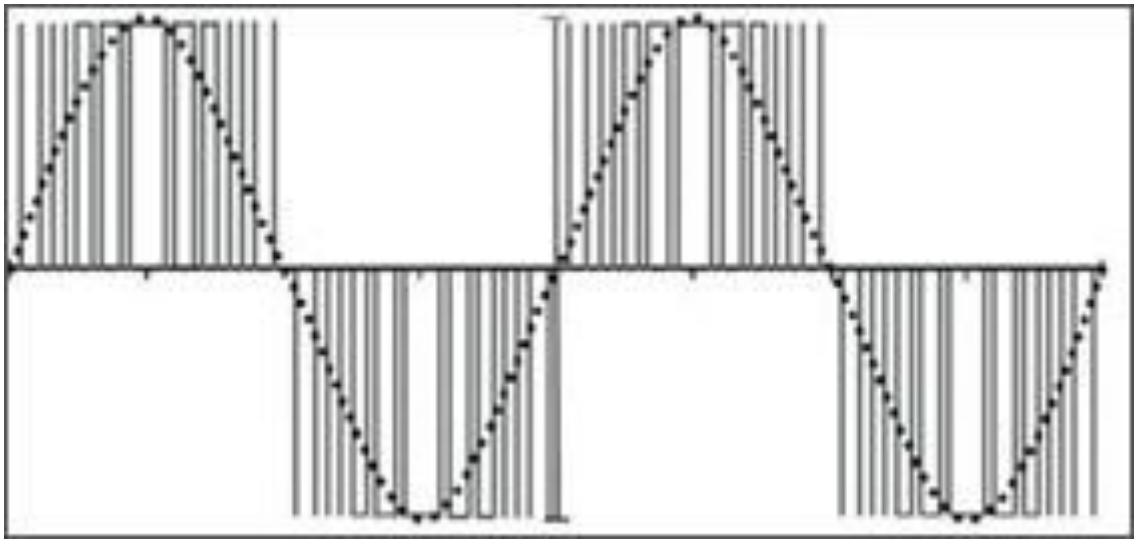
3.1 Taajuusmuuttajan toiminta

Molemmat kolmi- ja yksivaihetaajuusmuuttajat voivat muuttaa sisään tulevaa tehoa mukauttamalla taajuutta ja jännitettä. Jännitteen taajuus määrää moottorin nopeuden.[10] Kaavan 2 perusteella voidaan laskea moottorin nopeuden.

$$N = 120 * \frac{f}{p} \tag{2}$$

[7] Danfoss, Mikä on taajuusmuuttaja?, 2023
[8] Danfoss, Mikä on taajuusmuuttaja?, 2023
[9] Danfoss, Mikä on taajuusmuuttaja?, 2023
[10] Danfoss, Mikä on taajuusmuuttaja?, 2023

Taajuusmuuttajan avulla sisään tulevasta tasavirrasta saadaan vaihtovirtaa käyttämällä pulssinleveysmodulaatiota vaihtamalla kytkimien asentoa eri ajanhetkellä, joka tuottaa keskiarvotettuna siniaaltoa muistuttavan jännitteen, kuten kuviossa 4 on esitettynä.



Kuvio 4. Pulssinleveysmodulaatiosta siniaalloksi.[11]

3.2 Taajuusmuuttajan energiasäästöt

Taajuusmuuttajan avulla saadaan useisiin sovelluksiin merkittäviä energiasäästöjä, joista yleisimpiä sovelluksia ovat pumput ja puhaltimet. Säädettävällä virtausmenetelmällä voidaan muuttaa virtauskäyrää ja vähentää merkittävästi tehon tarvetta. Keskipakolaitteet, kuten puhaltimet, pumput ja kompressorit noudattavat yleisiä nopeuden lakeja. Nämä lait määrittelevät suhteet nopeuden ja muuttujien virtaus, paine ja teho välillä. [12]

[11] Variable Frequency Drives, Variable Frequency Drives, 2023
[12] Variable Frequency Drives, Variable Frequency Drives, 2023

Useimmat moottorit on suunniteltu pyörimään tietyllä nopeudella, joka perustuu niiden sisään rakennettujen magneettinapojen lukumäärään ja käytettävään syöttöjännitteeseen ja -taajuuteen. Ilman säätömahdollisuuksia moottoreiden nopeuksia ei voida muuttaa. Lisäksi on vaikea löytää juuri sopivalla nopeudella pyörivää moottoria haluttuun sovellukseen. Näin ollen säädettävät moottorit mahdollistavat tehokkaan virrankäytön ja prosessinhallinnan, sekä vähentää koneiden kulumista ja lisäävät tehokerrointa. [13]

[13] Control Techniques, How Variable Speed Drives Save Energy?, 2023

4 INDUKTIOMOOTTORI

Induktiomoottori (tunnetaan myös epäsynkroninen, sekä oikosulkumoottori) on yleisesti käytetty vaihtovirtamoottori. Induktiomoottorissa vääntömomentin tuottamiseen tarvittava roottorin sähkövirta saadaan sähkömagneettisen induktion kautta staattorikäämin pyörivästä magneettikentästä. [14]

Useimmat induktiomoottorit ovat suunniteltuja toimimaan kolmivaihejännitteellä. Taajuusmuuttajien lähteenä on yleensä vaihtosuuntaaja, joka käyttää puolijohde kytkimiä tuottaakseen sinimuotoista jännitettä ja virtaa kontrolloidakseen taajuutta. [15]

Kolmivaihe moottoreita käytetään huomattavasti useammin teollisuudessa, sillä niiden tehon tarve on suurempi kuin yksi vaiheisen moottorin [16]. Induktiomoottori on ylivoimaisesti yleisin tänä päivänä käytössä oleva sähkömoottori. Tämä johtuu siitä, että ne ovat rakenteeltaan yksinkertaisia, suhteellisen halpoja valmistaa ja erittäin luotettavia, sillä niissä on käytännössä vain yksi liikkuva osa. Toisin kuin esimerkiksi tasavirtamoottoreissa tai yleismoottoreissa on harjoja ja liukukoskettimia. [17]

[14] Electrical4U, Induction Motor: Working Principle, Types & Definition, 2023
[15] Bose B. K., Power Electronics and Variable Frequency Drives, 1996, 42
[16] Duke Electric, Induction Motors: What Are They, How do They Work, and How Are They Used?, 2023
[17] Circuit Cellar, Induction Motors, 2023

5 ARM-OHJAIN

Mikrokontrolleriyksikkö (Micro controller unit) on älykäs puolijohdesiru, joka koostuu prosessoriyksiköstä, muistimoduuleista, kommunikaatioväylistä, I/O-pinneistä sekä lisälaitteista. Mikrokontrollereita käytetään laajasti erilaisissa sovelluksissa, kuten pesukoneissa, roboteissa, droneissa, radioissa ja lukuisissa teollisuuden laitteistoissa.[18]

Vaikka mikrokontrollerissa on prosessoriyksikkö, sen arvo ei rajoitu pelkästään binääristen arvojen laskemiseen. Sen todellinen arvo on sen kyvyssä liittää fyysinen maailma yhteen sen sisäänrakennettujen kommunikaatio- ja oheislaitteiden avulla. [19]

Teknisesti mikrokontrolleri toimii suorittamalla ohjelman sen ”non-volatile” muistimoduuliin tallennetuista ohjeista. Aikaisemmin kontrollerit perustuivat ROM-tekniikkaan, joten ohjelmadata oli vaikea tai mahdoton poistaa. Flash-tekniikan mullistaessa puolijohdetekniikkaa, MCU:t alkoivat tallentaa ohjelman ohjeita sisäänrakennettuun flash-muistiin. [20]

Useimmat nykyaikaiset mikrokontrollerit käyttävät RISC (Reduced Instruction Set Computer) -ohjeiden arkkitehtuuria perusohjeiden käsittelyyn. Sulautettujen järjestelmien ohjelmoimiseen käytetään tänä päivänä pääosin C-, assembler ja C++-ohjelmointikieliä. [21]

[18] Cadence, What is an MCU and How do Microcontroller Units Work, 2023
[19] Cadence, What is an MCU and How do Microcontroller Units Work, 2023
[20] Cadence, What is an MCU and How do Microcontroller Units Work, 2023
[21] Cadence, What is an MCU and How do Microcontroller Units Work, 2023

5.1 Ajastimet

Sulautetuissa järjestelmissä ajastimia käytetään mittaamaan tiettyjä aikavälejä. Hyvin usein ajastimia kutsutaan myös laskureiksi. Ajastin on komponentti, jota käytetään laajalti erilaisissa sulautetuissa järjestelmissä. Niitä käytetään ajan tallentamiseen eri tapahtumille tai esimerkiksi mittaamaan aikaa tapahtumien välillä. Ajastin on yksinkertaisempi binäärilaskuri, joka on konfiguroitu piirissä tai järjestelmässä laskemaan järjestelmän pulssit tarpeen mukaan. [22]

Ajastimia voidaan säätää tarpeen ja sovelluksen mukaan. Esimerkiksi ajastin voidaan ohjelmoida toimimaan niin, että ajastimen saavuttaessa maksimiarvonsa, laitteisto luo niin sanotun ylivuotolipun ja ajastimen arvo palautetaan takaisin nollaan. Näin ollen ajastinta voidaan käyttää mittamaan kulunutta aikaa tai tietyllä aikavälillä tapahtuvia ulkoisia tapahtumia. Ajastimen tahdistus otetaan joko laitteiston sisäisestä tai ulkoisesta kellotaajuudesta. Tämän lisäksi ajastimen avulla voidaan luoda kellopulsseja, joita käytetään sarjaviestinnässä.[23]

5.2 Keskeytykset

Keskeytys on laitteiston tai ohjelmiston lähettämä signaali prosessorille tapahtumasta, joka vaatii välitöntä huomiota. Käytännössä on mahdollisuus käyttää kolmea erilaista keskeytystä: laitteisto, ohjelmisto tai ”kysely”. Kun keskeytys tapahtuu, laitteisto suorittaa viimeisimmän komennon loppuun aloittaa keskeytyspalvelurutiinin (ISR, Interrupt Service Routine) tai keskeytyksen käsittelyn. Keskeytyspalvelurutiini kertoo prosessorille tai ohjaimelle mitä keskeytyksen tapahtuessa tehdään. [24]

---

[22] Electgo, Timers | All you need to know: Types, Operating Mode, Applications, 2019
[23] Electgo, Timers | All you need to know: Types, Operating Mode, Applications, 2019
[24] Tutorialspoint, Embedded Systems - Interrupts, 2023

Laitteistokeskeytys on hälytyssignaali, joka lähetetään prosessorille ulkoisesta laitteesta, kuten levyohjaimesta tai ulkoisesta lisälaitteesta. Esimerkiksi kun painamme näppäimistön näppäintä tai liikutamme tietokoneen hiirtä, ne laukaisevat laitteistopohjaisen keskeytyksen ja prosessori aloittaa lukemaan painetun näppäimen tai hiiren sijainnin. [25]

Ohjelmistokeskeytyksen aiheuttaa joko haluttu tila tai erityskäsky, joka aiheuttaa keskeytyksen prosessorille. Esimerkiksi, jos prosessori suorittaa käskyn jakaa luku nollalla ja nolla aiheuttaa keskeytyksen, näin täyttyy ehdot keskeytykselle. [26]

Jatkuvan seurannan tilat tunnetaan yleisesti kyselynä. Mikrokontrolleri seuraa jatkuvasti esimerkiksi oheislaitteen tilaa ja tilan muuttuminen voidaan ohjelmoida aiheuttamaan keskeytys. Kyselykeskeytystä käytetään yleensä tilanteissa, jolloin ei haluta käyttää kaikkea ohjelmiston aikaa yksittäisen tilan seuraamiseen vaan tilan muutos aiheuttaa keskeytyksen. [27]

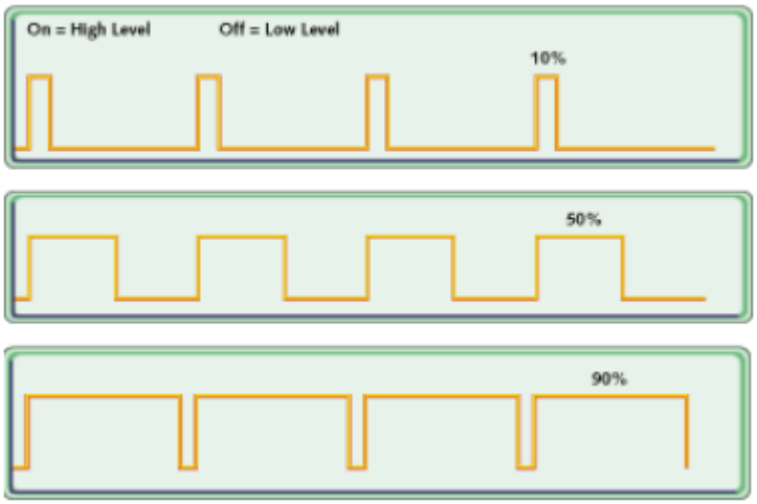
5.3 Pulssinleveysmodulaatio

Pulssinleveysmodulaatio (PWM) on tehokas tekniikka analogisten piirien ohjaamiseen mikroprosessorin digitaalisilla ulostuloilla. Kyseistä modulaatiota käytetään laajasti erilaisissa sovelluksissa, kuten mittauksissa, kommunikaatiossa, tehon ohjauksessa ja muuntamisessa. Käyttäen pulssinleveysmodulaatiota voidaan ohjata järjestelmien energiankulutusta merkittävästi. Lisäksi useat mikrokontrollerit sisältävät valmiina PWM-ohjaimen, mikä tekee tekniikasta helpon toteuttaa ja käyttää.[28]

[25] Tutorialspoint, Embedded Systems – Interrupts, 2023
[26] Tutorialspoint, Embedded Systems – Interrupts, 2023
[27] Tutorialspoint, Embedded Systems – Interrupts, 2023
[28] Barr M, Pulse Width Modulation (PWM): An Overview, 2001

Pulssinleveysmodulaatio on lyhyesti sanottuna tapa koodata analogiset signaalitasot digitaalisesti. Korkean resoluution laskureiden avulla neliöaallon pulssisuhde määritetään vastaamaan tiettyä analogista signaalitasoa. Pulssinleveysmodulaatio on kuitenkin digitaalinen, koska minkä tahansa hetken aikana koko tasavirta on täysin päällä tai pois päältä. Jännite- tai virtalähde toimitetaan analogiselle kuormalle sarjalla toistuvia päälle ja pois pulsseja. [29]

Kuviossa 5 on modulaatio, jossa 10 %, 50 % ja 90 % pulssisuhteet. Käytännössä tämä tarkoittaa esimerkiksi 10 % tilanteessa, että jännite on päällä vain 10 % ajasta ja muun ajan pois päältä.



Kuvio 5. Pulssinleveysmodulaatio eri pulssisuhteilla.[30]

[29] Barr M, Pulse Width Modulation (PWM): An Overview, 2001
[30] Barr M, Pulse Width Modulation (PWM): An Overview, 2001

6 PULSSINLEVEYSMODULAATIO VAIHTOVIRRAKSI

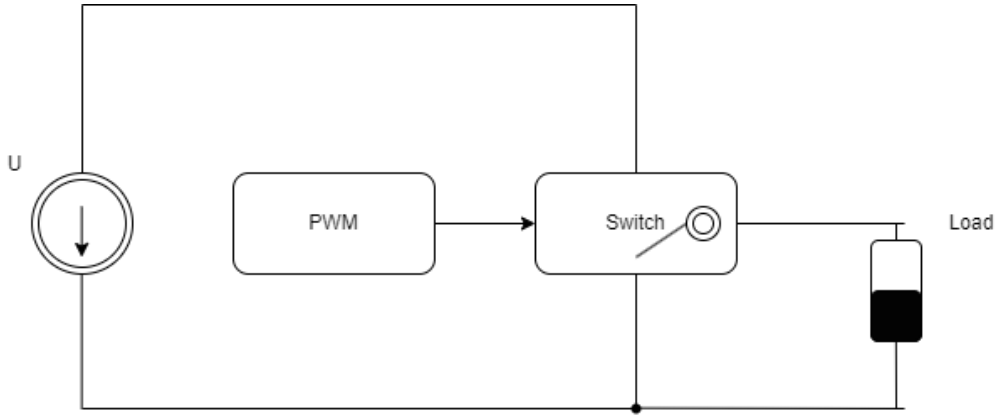
Tehokas ja nopea tapa kontrolloida teho elektroniikkaa on avain asemassa modernissa teollisuudessa. Se on toteutettu käyttäen elektronisia tehomuuntimia. Muuntimet siirtävät energiaa lähteestä kontrolloidussa prosessissa, jossa käytetään puolijohde kytkimiä. Kytkimiä säädetään päälle ja pois nopealla tahdilla. Algoritmi, joka toteuttaa vaihto funktion on pulssinleveysmodulaatio. [31]

Suurin osa teollisuudessa käytössä olevista moottoreista toimii vaihtelevissa nopeuksissa. Nämä laitteet vaativat muuttuvaa jännitettä sekä taajuutta kolmivaiheiselta syötöltä. Kun roottorin taajuus säädetään syöttötaajuuden avulla, koneen virtaus määritetään syöttöjännitteellä. Tehovaatimukset vaihtelevat muutamasta kilowatista useisiin megawatteihin. Yleensä teho otetaan tasavirtalähteestä ja muunnetaan kolmivaiheiseksi vaihtovirraksi käyttäen elektronisia DC-AC-muuntajia. Sisään tuleva tasavirta, joka on pääosin vakio, voidaan ottaa sähköyhtiöltä suuntaamalla. Vaihtoehtoisesti teho voidaan ottaa akustosta, tämä koskee keskeytymättömiä virtalähteitä ja sähköajoneuvoja.[32]

[31] Bose B. K., Power Electronics and Variable Frequency Drives, 1996, 138

[32] Bose B. K., Power Electronics and Variable Frequency Drives, 1996, 138

Kuviossa 6 on esitettynä miten pulssinleveysmodulaatio kytkettynä kytkimeen saadaan tuotettua tasavirrasta vaihtovirtaa. Modulaatio muuntaa jatkuvan syöttösignaalin kytkentähetkien sekvenssiksi. Kappaleessa 7.1 simulaatio on toteutettu kuvion 6 mukaisilla kytkennöillä. [33]



Kuvio 6. DC-to-AC kytkimellä.

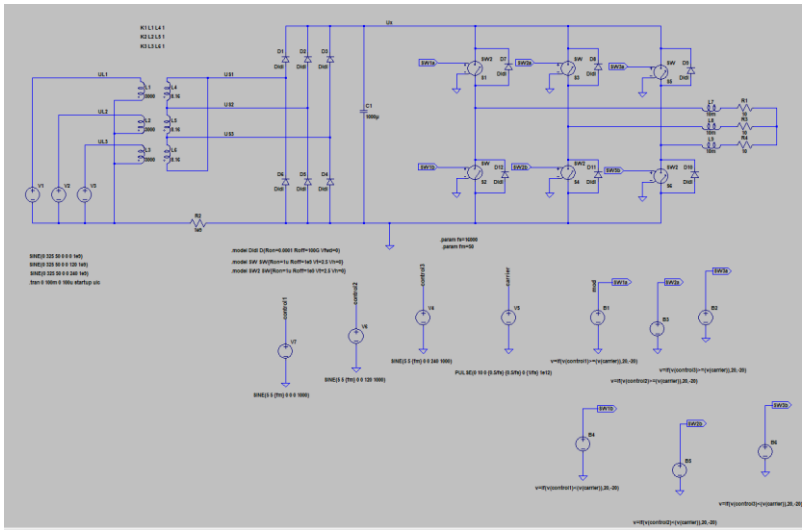
[33] Bose B. K., Power Electronics and Variable Frequency Drives, 1996, 138-139

7 TYÖN TOTEUTUS

Opinnäytetyön tavoitteena oli tutkia tapoja toteuttaa taajuusmuuttajan modulaattori, sekä tavan löydyttyä ohjelmoida ja testata kyseinen toteutus. Työssä käytettiin NUCLEO-l152re -kehitysalustaa, jolle ohjelmoitiin työssä tarvittava ohjelmisto. Opinnäytetyön tuotoksena saatiin nopea ja edullinen tapa toteuttaa taajuusmuuttajan modulaattori, jolla voidaan esitellä esimerkiksi opiskelijoille taajuusmuuttajan modulaattorin ja sähkömoottorien toimintaa.

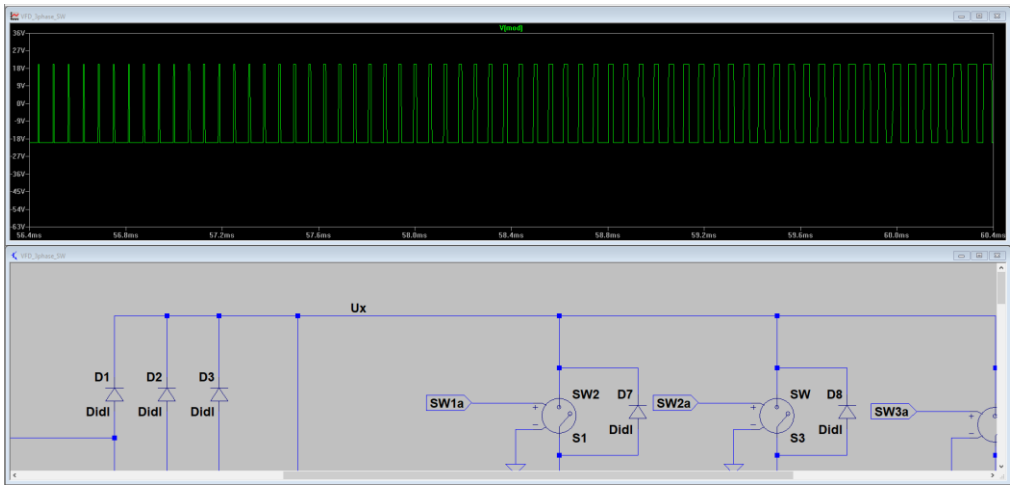
7.1 Simulaatiot

Työ aloitettiin tutkimalla Vaasan ammattikorkeakoulun opettajan toteuttamaa simulaatiota. Simulaatio toimi pohjana sekä teoriana toteutettavalle ohjelmistolle. Kuviossa 7 on esitettynä LTspice simulaatio, jossa jännitelähde, kytkimet, PWM-ajurit sekä ulostulo. Simulaatiossa oli kuusi kytkintä ja kolme vaihetta.



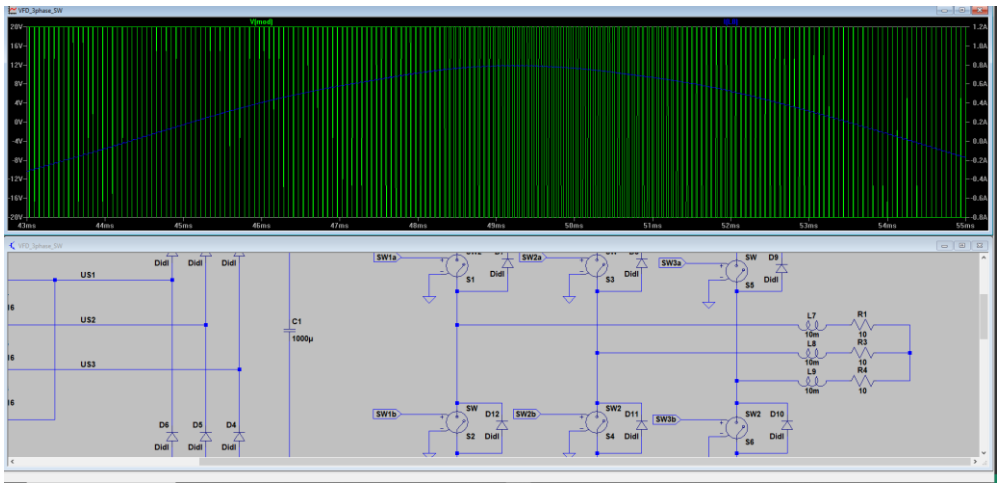
Kuvio 7. Simuloitu taajuusmuuttaja.

Kuviossa 8 on esitettynä SW1a syötettyä pulssinleveysmodulaatiota, tämä pulssi säätelee kytkimen asentoa. Simulaatio toteutettiin vastaamaan 50Hz siniaaltoa.



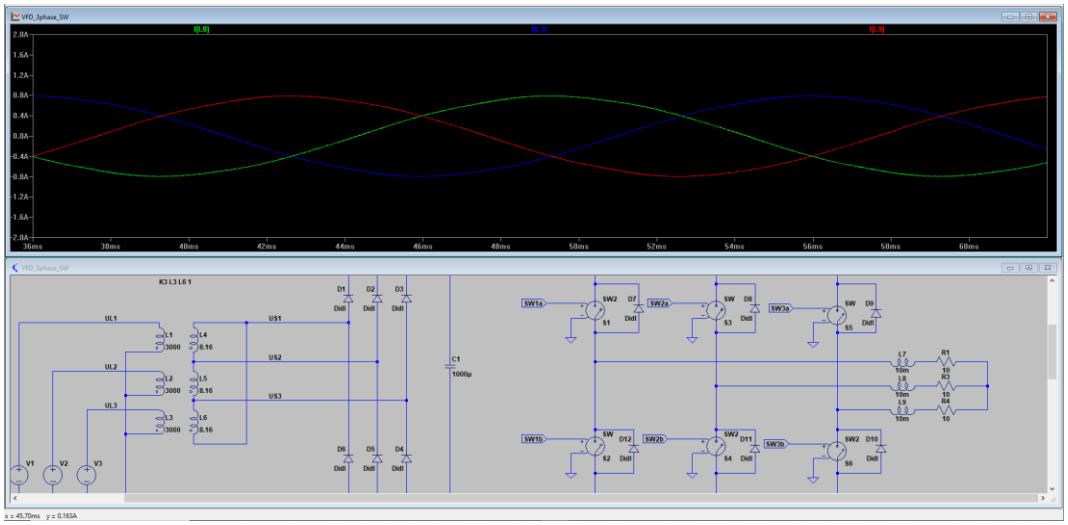
Kuvio 8. Pulssinleveysmodulaatio simuloituna kytkimelle.

Kuviossa 9 on esitettynä simulaatiosta yksittäisen vaiheen siniaalto sekä pulssinleveysmodulaatio.



Kuvio 9. Pulssinleveysmodulaatio sekä siniaalto käämiltä.

Kuviossa 10 on esitettynä kaikki kolme siniaaltoa, joiden vaihe erona on 120 astetta.



Kuvio 10. Kolme eri vaihetta siniaaltona.

Simulaatiot olivat onnistuneita ja näiden pohjalta pystyttiin toteamaan, että kuudella kytkimellä pystytään toteuttamaan kolme vaihetta.

7.2 PWM-arvojen määritys

Pulssinleveysmodulaation arvot määritettiin valmiiksi käyttämällä kaavaa 3. Näin säästettiin prosessorilta aikaa ja resursseja ajon aikana. Laskelmat tehtiin etukäteen taajuuksille 5-50 Hz, jolloin pystyttiin toteuttamaan myös induktiomoottorin vaatiman kiihdytyksen täyteen vauhtiin.

$$y(t) = A * \sin(2\pi f t + \varphi) \quad (3)$$

A = amplitudi

f = taajuus

t = ajan hetki

φ = vaihe

Taulukossa 1 on esitettynä ensimmäiset 15 arvoa esimerkkinä ajastimelle lasketuista arvoista. Alla olevat laskut pohjautuvat 16 bittisiin arvoihin. Prosessorin taajuus on 32 Mhz ja automaattinen latausrekisteri 16000. Korkeampaan tarkkuuteen olisi vaatinut huomattavasti korkeampi taajuuksinen prosessori.

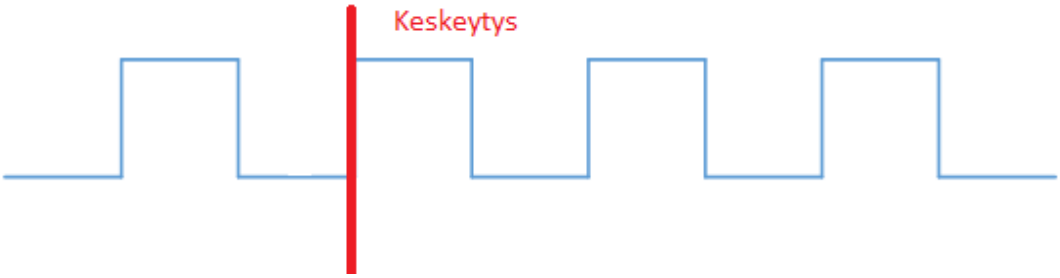
Taulukko 1. Ensimmäiset 15 PWM-arvoa.

nro	aika (t)	16 bit pwm 50 Hz
1	0,00006250	1287
2	0,00012500	2573
3	0,00018750	3858
4	0,00025000	5142
5	0,00031250	6424
6	0,00037500	7703
7	0,00043750	8979
8	0,00050000	10252
9	0,00056250	11521
10	0,00062500	12785
11	0,00068750	14045
12	0,00075000	15299
13	0,00081250	16547
14	0,00087500	17789
15	0,00093750	19024

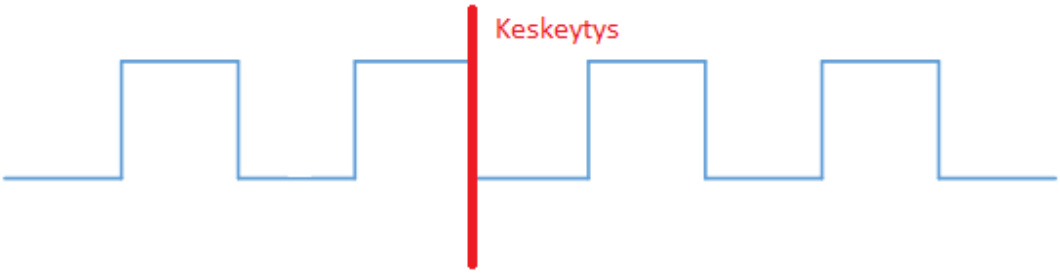
7.3 Ohjelmointi

Taajuusmuutajan modulaattorin ohjelmakoodi toteutettiin C-kielellä. Itse pääohjelma ei sisällä juurikaan muuta toiminnallisuutta kuin ajastimien käynnistyksen sekä taajuuksien muuttamisen moottorin käynnistämistä varten. Kaikista suurimman työn vaati pulssinleveysmodulaatioita varten ohjelmoitujen ajastimien käyttö, sekä keskeytykset, jotka muuttavat modulaation pulssisuhdetta jokaisen pulssin jälkeen. Lisäksi erityisen tärkeää oli löytää oikeat rekisterien arvot, jotta pystyttiin toteuttamaan keskeytykset nousevalta ja laskevalta reunalta.

Kuvioissa 11 ja 12 on havainnollistettu keskeytyksen ajoittaminen nousevalle ja laskevalle reunalle. Keskeytyksen tapahduttua muutetaan jokaisen ajastimen arvoa seuraavaan laskettuun arvoon.



Kuvio 11. Keskeytys nousevalta reunalta.



Kuvio 12. Keskeytys laskevalta reunalta.

7.4 Testaus

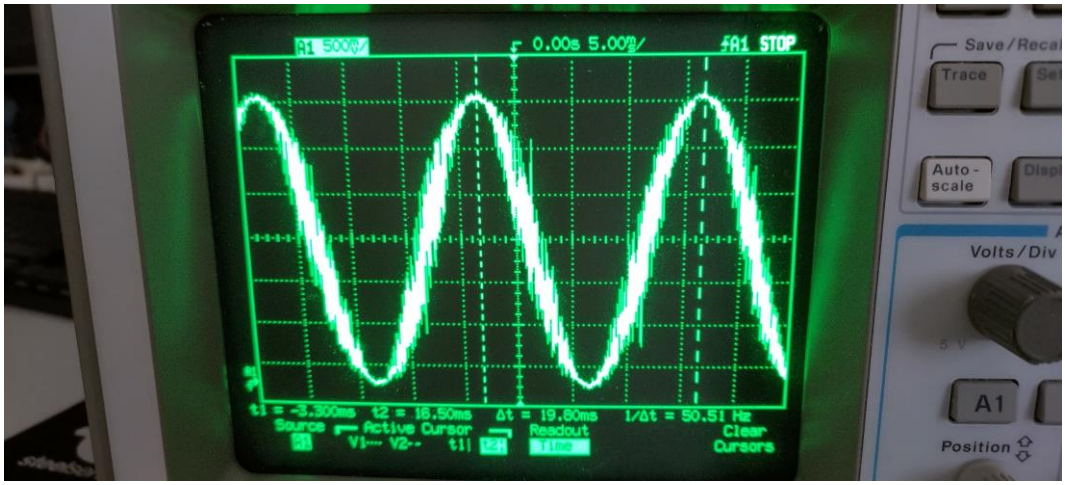
Ohjelmiston pulssinleveysmodulaatiota testattiin käyttämällä alipäästösuodatinta, joka laskettiin käyttäen kaavaa 4 Tämän avulla saatiin oskilloskoopin kautta tutkittua onko tuotettu siniaalto haluttua sekä onko taajuus oikea.

$$\frac{1}{2\pi RC} \tag{4}$$

R = Resistanssi

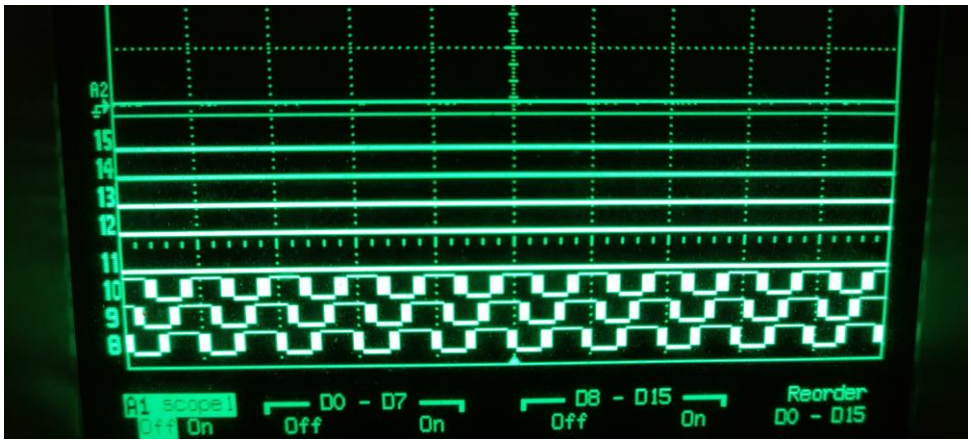
C = Kapasitanssi

Kuvassa 1 on oskilloskoopilla mitattuna 50 Hz taajuus yhdestä vaiheesta, joka on odotetun näköinen. Kuvasta voidaan päätellä ohjelmoinnin toimineen odotetulla tavalla.



Kuva 1. 50 Hz siniaalto.

Kuvassa 2 on kaikki kolme toteutettua vaihetta digitaalisena, jolloin saatiin yhteen näkymään jokainen vaihe. Kuvasta nähdään helposti vaihe ero, joka ohjelmoitiin olemaan haluttu 120°**.**



Kuva 2. Kolme vaihetta digitaalisena.

8 OHJELMAKOODI

Seuraavissa kappaleissa käydään läpi työn ohjelmakoodin tärkeimmät funktiota ja niiden toiminnot. Tärkeintä ohjelmoitaessa olivat ajastimet, keskeytykset sekä etukäteen lasketut

8.1 Ajastimien alustaminen

Funktio init_PWM_TIM2 alustaa PWM-arvot ajastimelle TIM2, GPIOA pinni 5 toimii ulostulona. Funktiossa määritetään esijakaja, automaattinen uudelleenlatausrekisteri, laskuri ja ajastimen toiminta PWM-tilana. Alustuksen pohjalta muodostuu kytkentätaajuudeksi 16 000 kHz. Lisäksi ajastin on 16 bittinen. Tämä funktio toteutettiin TIM3 ja TIM4 ajastimille samanlaisena.

Lisäksi se asettaa TIM2:n keskeytyslipun laukeamaan laskevalla reunalla ja ottaa käyttöön TIM2-keskeytyksen NVIC:ssä (Nested Vectored Interrupt Controller) ajastinkeskeytyksen käsittelemiseksi.

```
void init_PWM_TIM2(void) {
    /* Basic PWM setup */
    // AHB peripheral clock enable register
    RCC->AHBENR |= 1;                       // Enable GPIOA
clock
    // GPIO alternate function low register (GPIOx_AFRL)
    // GPIOA->AFR[0] |= 0x00100000;         // PA5 pin for
tim2
    GPIOA->AFR[0] |= (1 << 20);
    // GPIO port mode register (GPIOx_MODER) (x = A..H)
    GPIOA->MODER &= ~0x0000C00;             // Clear bits
    GPIOA->MODER |= (1 << 11);              // Set bits

    //Setup TIM2
    // APB1 peripheral clock enable register
    RCC->APB1ENR |= 1;                      // Enable TIM2
clock
    // TIMx prescaler (TIMx_PSC)
```

```
    TIM2->PSC = TIM2_PRESCALER_VAL - 1;      // divided by 1
= 32000000
    // TIMx auto-reload register
    TIM2->ARR = TIM2_ARR_REGISTER - 1;       // divided by
2000 = 16000
    // TIMx counter (TIMx_CNT)
    TIM2->CNT = 0;
    // TIMx capture/compare mode register 1
    TIM2->CCMR1 = 0x0060;                    // PWM mode
    // TIMx capture/compare enable register (TIMx_CCER)
    TIM2->CCER = 1;                          // Enable PWM
Ch1
    // TIMx capture/compare register 1
    TIM2->CCR1 = TIM2_DUTY_CYCLE - 1;        // Pulse width
1/3 of the period
    /************************/

    /* Polarity */
    // Bit 1 CC1P: Capture/Compare 1 output Polarity
    // TIM2->CCER |= (1 << 1);
    /**************/

    /* Interrupts */
    TIM2->DIER |= (1 << 1);                  //enable UIE,
interrupt enable -> falling edge
    // // TIM2->DIER |= 1;                  //enable UIE,
interrupt enable -> interrupt from ccr1 val
    NVIC_EnableIRQ(TIM2_IRQn);
    /**************/
}
```

8.2 Keskeytykset

Työssä keskeytykset olivat keskeisessä roolissa, sillä niiden avulla pystyttiin muuttamaan modulaation pulssisuhdetta jokaisen syklinä jälkeen. Keskeytystä kutsuttiin kappaleen 7.3 kuvioissa 11 ja 12 mukaisella tavalla, kun siniaalto oli positiivisella puolella keskeytystä kutsuttiin nousevalta reunalta ja taas negatiivisella puolella laskevalta reunalta. Keskeytyksiä tuli siis jokaisen syklin jälkeen.

Keskeytysfunktiossa TIM2_IRQHandler hyödynnettiin globaalia muuttujaa ramp_up_counter määrittämään moottorin käynnistysvaiheen, jossa taajuutta nostettiin 5 Hz kerrallaan aina 50 Hz täyteen vauhtiin saakka. Laskurin tim2_counter avulla käytiin läpi taulukkoa, jossa oli pulssisuhteen arvot. Siniaallon puolessa välissä CCER rekisterin arvolla modulaation polariteetti käännettiin, jonka avulla pystyttiin toteuttaa siniaallon negatiivinen puoli. TIM2_IRQHandler funktio toteutettiin samanlaisena kaikille kolmelle ajastimelle, mutta pulssisuhteen arvot olivat erilaiset, jotta pystyttiin toteuttamaan 120° vaihe-ero.

```
void TIM2_IRQHandler(void) {
    TIM2->SR=0;                             //clear UIF
    // GPIOA->ODR ^= (1 << 8);
    // set new duty value
    if (ramp_up_counter == 0) {
        TIM2->CCR1 = phase1_table_5hz[tim2_counter];
        if (tim2_counter == LAST_PWM_VAL_5Hz) {
            tim2_counter = 0;
            // change polarity
            TIM2->CCER ^= (1 << 1);
        }
    }
    if (ramp_up_counter == 1) {
        if (tim2_counter > LAST_PWM_VAL_10Hz) {
            tim2_counter = 0;
        }
        TIM2->CCR1 = phase1_table_10hz[tim2_counter];
        if (tim2_counter == LAST_PWM_VAL_10Hz) {
            tim2_counter = 0;
            // change polarity
            TIM2->CCER ^= (1 << 1);
        }
    }
    if (ramp_up_counter == 2) {
        if (tim2_counter > LAST_PWM_VAL_15Hz) {
            tim2_counter = 0;
        }
        TIM2->CCR1 = phase1_table_15hz[tim2_counter];
        if (tim2_counter == LAST_PWM_VAL_15Hz) {
```

```
            tim2_counter = 0;
            // change polarity
            TIM2->CCER ^= (1 << 1);
        }
    }
    if (ramp_up_counter == 3) {
        if (tim2_counter > LAST_PWM_VAL_20Hz) {
            tim2_counter = 0;
        }
        TIM2->CCR1 = phase1_table_20hz[tim2_counter];
        if (tim2_counter == LAST_PWM_VAL_20Hz) {
            tim2_counter = 0;
            // change polarity
            TIM2->CCER ^= (1 << 1);
        }
    }
    if (ramp_up_counter == 4) {
        if (tim2_counter > LAST_PWM_VAL_25Hz) {
            tim2_counter = 0;
        }
        TIM2->CCR1 = phase1_table_25hz[tim2_counter];
        if (tim2_counter == LAST_PWM_VAL_25Hz) {
            tim2_counter = 0;
            // change polarity
            TIM2->CCER ^= (1 << 1);
        }
    }
    if (ramp_up_counter == 5) {
        if (tim2_counter > LAST_PWM_VAL_30Hz) {
            tim2_counter = 0;
        }
        TIM2->CCR1 = phase1_table_30hz[tim2_counter];
        if (tim2_counter == LAST_PWM_VAL_30Hz) {
            tim2_counter = 0;
            // change polarity
            TIM2->CCER ^= (1 << 1);
        }
    }
    if (ramp_up_counter == 6) {
        if (tim2_counter > LAST_PWM_VAL_35Hz) {
```

```
            tim2_counter = 0;
        }
        TIM2->CCR1 = phase1_table_35hz[tim2_counter];
        if (tim2_counter == LAST_PWM_VAL_35Hz) {
            tim2_counter = 0;
            // change polarity
            TIM2->CCER ^= (1 << 1);
        }
    }
    if (ramp_up_counter == 7) {
        if (tim2_counter > LAST_PWM_VAL_40Hz) {
            tim2_counter = 0;
        }
        TIM2->CCR1 = phase1_table_40hz[tim2_counter];
        if (tim2_counter == LAST_PWM_VAL_40Hz) {
            tim2_counter = 0;
            // change polarity
            TIM2->CCER ^= (1 << 1);
        }
    }
    if (ramp_up_counter == 8) {
        if (tim2_counter > LAST_PWM_VAL_45Hz) {
            tim2_counter = 0;
        }
        TIM2->CCR1 = phase1_table_45hz[tim2_counter];
        if (tim2_counter == LAST_PWM_VAL_45Hz) {
            tim2_counter = 0;
            // change polarity
            TIM2->CCER ^= (1 << 1);
        }
    }
    if (ramp_up_counter == 9) {
        if (tim2_counter > LAST_PWM_VAL_50Hz) {
            tim2_counter = 0;
        }
        TIM2->CCR1 = phase1_table_50hz[tim2_counter];
        if (tim2_counter == LAST_PWM_VAL_50Hz) {
            tim2_counter = 0;
            // change polarity
            TIM2->CCER ^= (1 << 1);
```

```
        }
    }

    // TIM2->CCR1 = phase1_table_50hz[tim2_counter];
    // if (tim2_counter == LAST_PWM_VAL) {
    //  tim2_counter = 0;
    //  // change polarity
    //  TIM2->CCER ^= (1 << 1);
    // }
    tim2_counter++;
}
```

8.3 Pääohjelma

Pääfunktiossa kutsutaan useita alustusfunktioita järjestelmäkellon sekä muiden ajastimien konfiguroimiseksi. Lisäksi ohjelmoinnin helpottamiseksi ajastimet pysäytetään testausvaiheessa. Alustusvaiheiden jälkeen siirrytään määritetyn napin painalluksella toteutusvaiheeseen. Ohjelma käynnistää kolme ajastinta ja jos DEBUG_PHASES-lippu on asetettu 1:een, ohjelma odottaa, että painiketta painetaan uudelleen ja korottaa tajuutta 5 Hz. Muussa tapauksessa ohjelma nostaa lähtötaajuuden 50 Hz:een 9 askeleessa, odottaen 1 sekunnin kunkin askeleen välillä.

```
int main(void) {
    /* Configure the system clock to 32 MHz and update
SystemCoreClock */
    SetSysClock();
    SystemCoreClockUpdate();
    USART2_Init();
    init_PWM_TIM2();
    init_PWM_TIM3();
    init_PWM_TIM4();
    init_TIM9_upcounting();
    init_GPIO_PA8();

    //debug_APB1_TIM2_unfreeze();
    debug_APB1_TIM2_freeze();
```

```
    debug_APB1_TIM3_freeze();
    debug_APB1_TIM4_freeze();
    debug_APB2_TIM9_freeze();

    // Start timers

    TIM9->CR1 = 1;

    RCC->AHBENR|=0x4; //GPIOC enable. p154
    GPIOC->MODER&=~0xC000000; //PC13 configured to input,
C=1100. p184
    while(1) {
        if(~(GPIOC->IDR) & 0x2000){
            TIM2->CR1 = 1;
            TIM3->CR1 = 1;
            TIM4->CR1 = 1;

            delay_Ms_osc(1000);
            while (1) {
                // debug Hz tables
                if (DEBUG_PHASES == 1) {
                    if(~(GPIOC->IDR) & 0x2000){
                        delay_ms(500);
                        ramp_up_counter++;
                            if (ramp_up_counter == 9) {
                                while(1); // full speed 50hz
                            }
                    }
                }
                else {
                    // ramp up 5 hz per 1 sec
                    ramp_up_counter++;
                    delay_Ms_osc(1000);
                    if (ramp_up_counter == 9) {
                        while(1); // full speed 50hz
                    }
                }


            }
```

```
        }
    }
    return 0;
}
```

8.4 Muut funktiot

Ohjelmakoodista löytyy myös useita muitakin funktioita, jotka ovat apuna pääohjelman toteutukselle. Funktiossa init_GPIO_PA8 alustetaan pääohjelamasta kutsuttua painonappia.

```
void init_GPIO_PA8(void) {
    RCC->AHBENR |= 1;                    // Enable PORT A
clock, does not matter if its already on
    GPIOA->MODER &= ~((1 << 17) | (1 << 16));       // Clear
bits 16,17
    GPIOA->MODER |= (1 << 16);           // bits 16. output
01.
}
```

Funktiossa init_TIM9_upcounting alustettiin ajastin odotusfunktiota varten. Tämän avulla voitiin tehdä odotusfunktio, joka odottaa annetun millisekuntien ajan.

```
void init_TIM9_upcounting(void) {
    // AHB peripheral clock enable register
    RCC->AHBENR |= (1 << 1);                        // Enable
GPIOC clock
    // GPIO alternate function low register (GPIOx_AFRL)
    GPIOC->AFR[1] |= (1 << 21) | (1 << 20);          // PC13
pin for tim9

    RCC->APB2ENR |= (2 << 1);                       // Enable
TIM9 clock

    TIM9->PSC = 3200 - 1;                          // divided
by 16000
```

```
    // TIMx auto-reload register
    TIM9->ARR = 65500 - 1;                          // divided
by 26667
    // TIMx counter (TIMx_CNT)
    TIM9->CNT = 0;
    // TIMx capture/compare mode register 1

}
```

Tämä funktio luo viiveen millisekunteina käyttäen laitteistopohjaista ajastinta mittaamaan kulunutta aikaa, sen sijaan että käyttäisi ohjelmistosilmukkaa.

```
void delay_ms(uint16_t wait) {
    wait *= 10;
    uint16_t difference = 0;
    uint16_t starttime = TIM9->CNT;
    while (difference <= wait) {
        difference = TIM9->CNT - starttime;
    }
}
```

Funktion calculate_phase1 avulla oltaisi voitu laskea ajon aikana pulssinleveysmodulaation vaatimat arvot, mutta tässä versiossa työtä sitä ei käytetty.

```
void calculate_phase1(void) {
    for (int i = 0; i < 160; i++) {
        test[i] = ((1000*sin(2*PI*i/320))+1000);
        // printf("%f %d\n", ((bits*sin(2*PI*i/320))+1000),
t);
    }

}
```

9 POHDINTA

Tämän opinnäytetyön tavoitteena oli löytää tapa toteuttaa sekä ohjelmoida taajuusmuuttajan modulaattori käyttäen ARM-ohjainta. Mielestäni tavoitteeseen päästiin ja saatiin toteutettua toimiva modulaattori. Jotta toteutus olisi voitu todeta myös oikean moottorin kanssa toimivaksi olisi täytynyt testata oikean laitteiston kanssa. Olisin halunnut toteuttaa tämän osan myös työstä, mutta ajan ja resurssien puutteen vuoksi tämä osa työstä jäi tekemättä.

Ohjelmistokoodin toteutus on välttävä prototyypiksi. Työhön olisi voinut lisätä esimerkiksi näppäimet, joilla nopeutta olisi voinut säätää. Lisäksi keskeytyksessä oleva koodi on hyvin pitkä, jolloin aikaa käytetään paljon keskeytyksessä. Tämä ei vaikuttanut lopputulokseen, mutta olisi mahdollista parantaa.

Työn loppuvaiheen puutteellisuuden vuoksi jatkotutkimuskohteena tulisi tutkia todellisuudessa laitteiston toimintaa toteutetulla ohjelmistolla. Lisäksi koodissa käytettiin huomattava määrä kovakoodattuja arvoja, joiden laskeminen olisi ajon aikana mahdollista toteuttaa nopeammalla prosessorilla. Yhtenä tutkimuskohteena voisi olla FPGA:lla toteutettu taajuusmuuttaja, jolloin pystyttäisiin toteuttamaan todella tarkka ja korkean taajuuden toteutus.

LÄHTEET

Barr, M. 2001. Pulse Width Modulation (PWM): An Overview. Viitattu 7.4.2023. https://www.embedded.com/introduction-to-pulse-width-modulation/

Bose, B. K. 1996. Power Electronics and Variable Frequency Drives. Piscataway. IEEE Press.

Cadence. 2023. What is an MCU and How do Microcontroller Units Work. Viitattu 7.4.2023.
https://resources.pcb.cadence.com/blog/2020-what-is-an-mcu-and-how-do-microcontroller-units-work

Circuit Cellar. 2023. Induction Motors. Viitattu 7.4.2023.
https://circuitcellar.com/resources/quickbits/induction-motors/

Control Techniques. 2023. How Variable Speed Drives Save Energy?. Viitattu 7.4.2023
https://acim.nidec.com/drives/control-techniques/downloads/how-drives-save-energy

Danfoss. 2023. Mikä on taajuusmuuttaja? Viitattu 7.4.2023.
https://www.danfoss.com/fi-fi/about-danfoss/our-businesses/drives/what-is-a-variable-frequency-drive/

Duke Electric. 2023. Induction Motors: What Are They, How do They Work, and How Are They Used? Viitattu 7.4.2023.
https://www.dukeelectric.com/blog/all-about-induction-motors-types-applications/

Electrical4U. 2023. Induction Motor: Working Principle, Types & Definition. Viitattu 7.4.2023
https://www.electrical4u.com/induction-motor-types-of-induction-motor/

Electgo. 2019. Timers | All you need to know: Types, Operating Mode, Applications. Viitattu 7.4.2023.
https://electgo.com/resources/timers

Gregersen, E. 2023. Alternating current, Viitattu 7.4.2023. https://www.britannica.com/science/alternating-current

National Fire Chiefs Council. 2023. The difference between a single-phase and a three-phase power systems. Viitattu 7.4.2023.
https://www.ukfrs.com/promos/17145

Tutorialspoint. 2023. Embedded Systems – Interrupts. Viitattu 7.4.2023
https://www.tutorialspoint.com/embedded_systems/es_interrupts.htm

Variable Frequency Drives. 2023. Variable Frequency Drives. Viitattu 7.4.2023.
http://www.vfds.org/

Wikipedia, 2023. Alternating current, Viitattu 7.4.2023.
https://en.wikipedia.org/wiki/Alternating_current

Wikipedia, 2023. Three-phase electric power. Viitattu 7.4.2023.
https://en.wikipedia.org/wiki/Three-phase_electric_power