Tampereen ammattikorkeakoulu

Trinamic TMC5160 -askelmoottoriohjaimen ohjainrajapinnan suunnittelu

Matti Saarela

OPINNÄYTETYÖ
Toukokuu 2020

Tieto- ja viestintätekniikan koulutus
Sulautetut järjestelmät ja elektroniikka

TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Tieto- ja viestintätekniikan koulutus
Sulautetut järjestelmät ja elektroniikka

SAARELA, MATTI:
Trinamic TMC5160 -askelmoottoriohjaimen ohjainrajapinnan suunnittelu

Opinnäytetyö 33 sivua
Toukokuu 2020

Työssä käsitellään Trinamic TMC5160 -askelmoottoriohjaimen liittämistä sulautettuun järjestelmään eritoten keskittyen sen ohjelmallisiin seikkoihin ohjainkirjastoa suunniteltaessa. Tässä työssä esitetään, kuinka TMC5160:n erityisominaisuudet saatiin valjastettua käyttöön esimerkkisovelluksessa.

Askelmoottoriohjaimen esimerkkisovelluksena käytetään työssä suunniteltua nelitahtimoottorin nopeussignaalisimulaattoria, jonka tehtävänä on mallintaa nykyaikaisten otto- ja dieselmoottorien moottorinohjauksen tarvitsemaa moottorinnopeussignaalia mekaanisesti. Simulaattorin toiminta perustuu askelmoottorin avulla pyöritettäviin rei'itettyihin kiekkoihin. Kiekkojen pyörimisliikkeestä reikien ja optisten antureiden avulla muodostetaan moottorinohjaimelle sopiva elektroninen signaali. Askelmoottorin ohjaamiseen käytetään TMC5160-askelmoottoriohjainta ja Arduino Due -kehityskorttia.

Simulaattorin ohjelmisto on kirjoitettu C-ohjelmointikielellä MISRA-C-tyyliohjeistusta mukaillen ja koostuu useasta kerroksesta. Työssä keskitytään tarkemmin matalimpaan ohjainkirjastokerrokseen, joka kommunikoi moottorinohjaimen kanssa.

Ohjainkirjaston rajapinta koostuu lähinnä funktioista, jotka kirjoittavat arvot vastaaviin askelmoottorin ohjaimen rekistereihin SPI-väylän välityksellä muuttaen ensin reaalimaailman arvot sopiviksi rekisteriarvoiksi. Sen lisäksi se sisältää alustusfunktion, jota täytyy kutsua, ennen kuin ohjainkirjastoa voi käyttää. Se asettaa TMC5160:n rekisterit sovellukselle suotuisiin alkuarvoihin.

Kehitettäessä ohjainta ja sen rajapintaa piti myös ottaa huomioon eräitä kyseisen askelmoottoriohjaimen rajoituksia. Tämän takia rajapintaan piti lisätä periodisesti suoritettavia erityisfunktioita, jotta rajoituksista johtuneet ongelmat saatiin kierrettyä. Kehitystyössä haastavinta oli sopivien alustusarvojen löytämien tasaisen pyörimisliikkeen saavuttamiseksi ja rekisteriarvojen muunnoksen optimointi kokonaislukulaskentaan sopivaksi.

Asiasanat: askelmoottori, moottorinohjausjärjestelmä

ABSTRACT

Tampereen ammattikorkeakoulu
Tampere University of Applied Sciences
Degree Programme in ICT-engineering
Embedded Systems and Electronics

SAARELA, MATTI:
Designing Driver Interface for Trinamic TMC5160 Stepper Motor Controller

Bachelor's thesis 33 pages
May 2020

This thesis introduces how to connect Trinamic TMC5160 stepper motor driver to an embedded system. It gives an insight into what is needed to implement TMC5160's special features in an application.

A mechanical four stroke engine speed signal simulator is used as the application for this work to mechanically simulate speed signals needed by electronic control units of modern petrol and diesel engines. Operation of the simulator is based on perforated discs driven by a stepper motor. Rotation of the discs is converted into electrical signals with optical interrupt sensors. The generated electrical signals can then be interpreted by an electronic control unit of a vehicle. TMC5160 stepper motor driver and Arduino Due prototyping board was used to drive the stepper motor.

Software of the simulator is written in C programming language after MISRA-C -style guide and consists of multiple software layers. The thesis concentrates on developing the lowest layer of the driver library, which handles communication between the stepper motor driver and the Arduino.

The Interface of the driver library mostly consists of functions that convert values in real world units to suitable register values and writes those to corresponding registers on stepper motor driver via SPI. It also includes an initialization function that needs to be called before driver interface can be used. It will set registers of TMC5160 to suitable initial values.

While developing the software driver and interface some limitations of the stepper motor driver were needed to take into account. Because of that some special periodically called workaround functions were introduced to avoid observed issues. Most challenging part of the development process was to find suitable initial register values to achieve smooth rotational operation and to optimize conversion formulas of the register values to be suitable for fast integer calculation.

Key words: stepper motor, engine control

SISÄLLYS

LYHENTEET JA TERMIT

Arduino	Avoimeen laitteistoon perustuva elektroniikka-alusta
CAN	Controller Area Network, automaatioväylä
FPGA	Field Programmable Gate Array, ohjelmoitava logiikka-piiri
H-silta	Yleensä DC-moottorien ohjaamiseen käytetty kytkentä
LCD	Liquid-crystal display, nestekidenäyttö
MISRA	Motor Industry Software Reliability Association, ajoneuvokäyttöön suunnatun ohjelmistokehityksen suosituksia kehittävä järjestö
MOSFET	Metallioksidi-puolijohdekanavatransistori
PWM	Pulse width modulation, pulssinleveysmodulaatio
SPI	Serial Peripheral Interface, yleensä laitteiden sisäiseen kommunikaatioon käytetty synkroninen sarjaliikenne-väylä
USB	Universal Serial Bus, oheislaitesarjaväyläarkkitehtuuri

1 JOHDANTO

Tässä työssä perehdytään mekaaniseen nelitahtimoottorin nopeussignaalisimulaattorin yhteydessä tehtyyn askelmoottoriohjaimen C-kieliseen rajapinta- ja ohjainkirjastoon. Työ tehtiin kesän 2018 aikana Wapice Oy:lle. Tämän yhteydessä perehdyttiin syvällisemmin askelmoottoreihin, niiden ohjaimiin ja erityisesti Trinamic Motion Control GmbH & Co. KG:n valmistamaan TMC5160-ohjaimeen, johon tässä opinnäytetyössä keskitytään tarkemmin.

Vaatimuksena oli suunnitella simulaattori, joka mallintaisi nelitahtisen otto- tai dieselmoottorin nopeussignaaleita mekaanisesti. Mekaanisella toiminnalla saataisiin signaaliin epälineaarisuuksia ja satunnaisuutta, joita oikean moottorin nopeussignaaleissakin esiintyy. Mekaanisesti tuotetusta signaalista saataisiin helposti luotettava myös moottorin käynnistyksen, pysäytyksen ja pyörimissuunnan vaihdon yhteydessä. Myös simulaattorin toimintaa on helpompi seurata visuaalisesti, kun signaali tuotetaan mekaanisella liikkeellä. Simulaattori suunniteltiin moottorinohjausjärjestelmien moottorin nopeuden ja nokka-akselin kulman eli moottorisyklin position tunnistuksen testaamiseen ja kehitykseen.

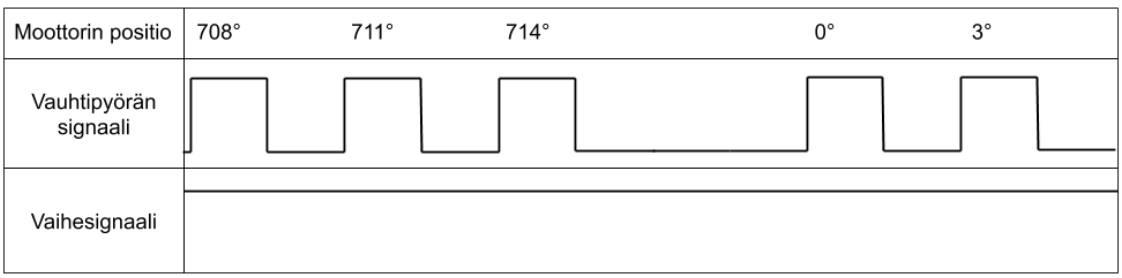
Rajapinta toteutettiin simulaattorissa käytetylle Atmel SAM3X8E -mikrokontrolleriin pohjautuvalle Arduino Due -kehitysalustalle. Kyseinen kehitysalusta valikoitui suuren liitäntäpinnien lukumäärän ja mikrokontrollerin hyvän suorituskyvyn takia.

2 MEKAANINEN NOPEUSSIGNAALISIMULAATTORI

2.1 Nelitahtimoottorin nopeussignaali

Nelitahtimoottorin syklissä on nimensä mukaisesti neljä toimintavaihetta: imutahti, puristustahti, työtahti ja poistotahti. Nämä neljä tahtia suoritetaan kahden kampiakselikierroksen aikana. Moottorin nopeutta mitataan kampiakseliin yhdistetyllä kiekolla tai vauhtipyörällä, johon on porattu reikiä tai koloja. Niiden lähelle on sijoitettu joko induktiivinen tai Hall-ilmiöön perustuva anturi, jonka avulla kiekon tai vauhtipyörän reiät tai kolot muunnetaan signaaliksi, jonka taajuus riippuu pyörimisnopeudesta. Nämä pulssit mahdollisesti siistitään jonkinlaisen komparaattorikytkennän avulla ja sen jälkeen syötetään moottorinohjausjärjestelmään. Reikiä tai koloja, joita usein kutsutaan hampaiksi, on vauhtipyörän tai kiekon ympärillä tasaisesti.

Kuitenkin jos moottorinohjausjärjestelmän täytyy tunnistaa tarkasti moottorin kulma eri ajan hetkillä, esimerkiksi suorasytytysjärjestelmän tai suoraruiskutusjärjestelmän ajoitusta varten, täytyy mitattavasta kiekosta pystyä määrittämään jokin moottorisyklin referenssipositio. Tämä referenssipositio on usein toteutettu puuttuvilla hampailla (kuva 1). Siinä ei jossain kiekon tai vauhtipyörän kohdassa olekaan yhtä tai useampaa hammasta paikassa, jossa niiden kuuluisi olla. Ohjausjärjestelmä tunnistaa tämän pulssijonon muutoksen referenssikulmaksi ja pystyy siten ankkuroimaan sisäisen moottorisyklin positiomittauksensa kyseiseen pisteeseen.

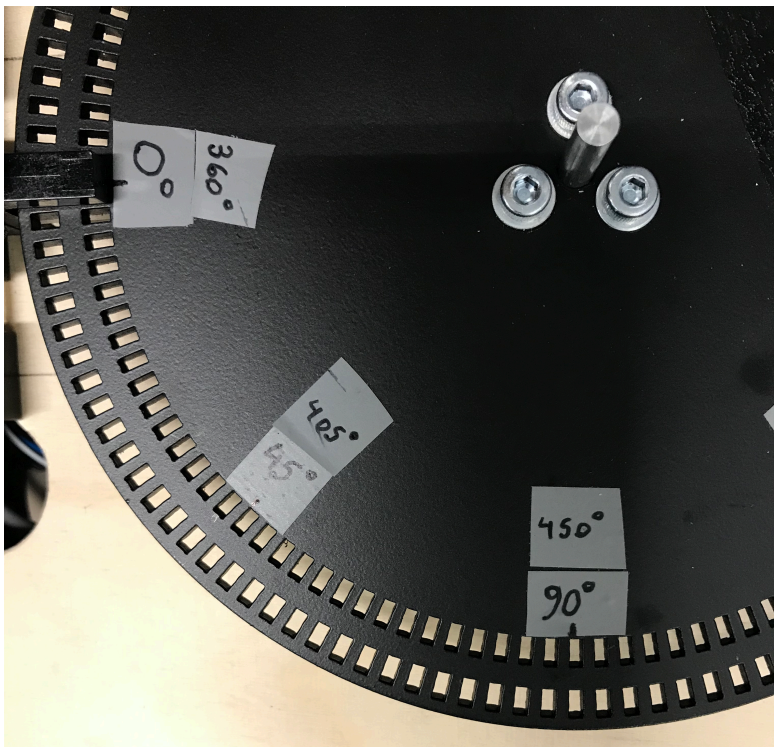


KUVA 1. Moottorin nopeussignaaliesimerkki 120-1-nopeussignaalilla.

Suunnitellussa nopeussignaalisimulaattorissa käytettiin 120-1-nopeussignaalia, eli simulaattorin akryylistä valmistetussa kiekossa on 120 laserleikattua sektoria,

joista yksi sektori on peitetty teipillä puuttuvaksi hampaaksi. Akryylinen kiekko on maalattu mustaksi siten, ettei se juuri läpäise valoa, sillä signaalit tuotetaan optisten keskeytysantureiden avulla (kuva 2). Kyseisten antureiden ulostulon jännite muuttuu, kun haarukka-anturin väliin asetetaan jotain, joka estää infrapunavalon pääsyn anturin toiselta puolelta toiselle puolelle.

Akryylikiekossa on kaksi riviä sektoreita ja kaksi anturia. Anturit on asetettu simulaattoriin 180 asteen päähän toisistaan. Toinen anturi tunnistaa sisemmän rivin sektoreita ja toinen ulomman rivin. Myös rivien puuttuvat hampaat ovat kiekon vastakkaisilla puolilla, joten anturien signaalit on mahdollista asettaa samanaikaisiksi.



KUVA 2. Nopeussignaalisimulaattorin kampiakselikiekko ja sisemmän reikärivin optinen anturi

2.2 Vaihesignaali

Kampiakselilta saatavan kulmatiedon perusteella ei voida vielä päätellä ollaanko nelitahtisessa syklissä imu- ja puristustahdin kierroksella, vai työ- ja poistotahdin kierroksella. Tähän tarvitaan nokka-akselilta saatavaa vaihetietoa. Nelitahtisen

moottorin nokka-akselit muun muassa avaavat ja sulkevat imu- ja pakoventtiileitä, jotka avautuvat vain kerran nelitahtimoottorin syklin aikana. Täten nokka-akseli pyörii puolta hitaammin kuin kampiakseli. Kampiakselilta saatavalla vaihetiedolla moottorinohjausjärjestelmä voi päätellä, kummalla syklin kierroksella moottori milläkin ajanhetkellä on.

Työssä suunnitellussa simulaattorissa on kampiakselin asentoa simuloivan kiekon akselista välitetty hammashihnan avulla toinen kiekko, joka pyörii puolet hitaammin. Tässä pienemmässä kiekossa on myös yksi samanlainen optinen haarukka-anturi, kuin suuremmassa kiekossa. Nokka-akselikiekossa toinen puoli ulottuu pidemmälle kuin toinen, joten joka toisella kampiakselikiekon puuttuvan hampaan kohdalla vaiheanturin ulostulossa näkyy jännite ja joka toisella ei.

2.3 Simulaattorin rakenne

Mekaaninen simulaattori (kuva 3) koostuu kampi- ja nokka-akselikiekosta ja yhteensä kolmesta optisesta anturista näiden reunoilla. Kaikki anturit ovat säädettäviä siten, että niiden positiota voidaan siirtää kiekkojen kehällä. Kampiakselikiekon toinen anturi on kiinnitetty pieneen ruuvipenkkiin, jolla suoritetaan tarkkoja anturin paikkamuutoksia. Muut anturit on kiinnitetty 3D-tulostettuun kiskoon ruuveilla, joten niiden tarkka säätäminen on hieman vaikeampaa.

KUVA 3. Mekaaninen nopeussignaalisimulaattori

Simulaattoriin kytketään 24 voltin tasajännite, joka pienennetään hakkurimoduuleilla 12 voltiksi nokka-akselikiekon optisen anturin käyttöjännitteeksi ja 5 voltiksi kampiakselin kiekon optisten antureiden ja Arduinon käyttöjännitteeksi. Arduinon käyttöjännitteen syöttöön käytettiin USB-kaapelia, sillä kyseisen kehityskortin version piirilevyssä on suunnitteluvirhe, josta johtuen piirilevyn käyttösähkön pääasiallinen liitin ei ole kytkettynä regulaattoriin lainkaan. Anturien ulostulosignaalit muunnetaan 0 - 24 voltin kanttiaalloksi H-sillan avulla, joten simulaattorin kaikki tulosignaalien ja lähtösignaalien jännitetasot ovat 24 volttia.

Simulaattoria voidaan ohjata etupaneelista inkrementaalisen enkooderin, painikkeiden ja LCD-näytön avulla, mutta tärkeämpää on ohjaus USB-sarjaliitännän kautta. Sen avulla käytössä ovat kaikki simulaattorin ominaisuudet ja simulaattorilla ajettavia testejä voidaan automatisoida.

Mikrokontrollerin sisäiseen flash-muistiin voidaan myös tallentaa nopeusnäytteitä, joiden avulla voidaan ajaa vaikkapa oikeasta moottorista taltioituja nopeusprofiileja.

2.4 Askelmoottoriohjain

Trinamic TMC5160 on monipuolinen askelmoottoriohjain, joka valittiin simulaattoriin suuren mikroaskelluskyvyn takia. Vaatimukset nopeussignaalisimulaattorin nopeuden maksimihuojunnalle ovat 10 % nopeusalueella 0,2 – 100 kierrosta minuutissa ja 1 % nopeusalueella 100 – 1000 kierrosta minuutissa. Kun valittua 200 fyysistä askelta sisältävää askelmoottoria pystytään askeltamaan 256 mikroaskeleella, saavutetaan 51 200 askelta kierrokselle, jolla saavutettiin vaatimukset täyttävä tasainen liike kaikilla vaadituilla nopeuksilla. TMC5160 myös mahdollistaa askelmoottorin kulman tunnistuksen kaikilla ajanhetkillä, joka on erittäin hyödyllinen ominaisuus simulaattorissa. Ohjaus TMC5160:lle tapahtuu SPI-väylän kautta. Tässä työssä käytettiin Arduinon valmista SPI-ohjelmistorajapintaa ohjelmiston kehitykseen kuluvan ajan säästämiseksi.

TMC5160-piiri ei itsessään sisällä askelmoottorin vaiheiden virtaa katkovia kanavatransistoreja, vaan TMC5160:n ulostulot on kytketty ohjaamaan ulkoisia kanavatransistoreita. Tämä mahdollistaa TMC5160:n käytön kaikenkokoisten askelmoottorien kanssa suunnittelemalla kytkentä käyttötarkoitukseen sopivilla kanavatransistoreilla.

Simulaattorin kehityksessä käytettiin TMC5160-EVAL-kehityskorttia, jonka kytkennässä on Infineon BSC072N08NS5 -MOSFET-kanavatransistoreita (TMC5160-EVAL BOM 2020). TMC5160-EVAL:n luvataan pystyvän syöttämään 4,7 ampeeria vaiheelle (TMC5160-EVAL tuotesivu 2020). Lopulliseen simulaattoriin valittiin 2 ampeerin vaihevirrankestolla varustettu askelmoottori, jonka todettiin kokeilemalla olevan enemmän kuin riittävä kyseiseen sovellukseen. Lopulliseen simulaattoriin valittiin fyysiseltä kooltaan pienempi ja pienempään 2,8 ampeerin vaihevirtaan pystyvä (TMC5160-BOB Datasheet 2020, 1) TMC5160-BOB-kehitysalusta. Kyseinen kortti käyttää askelmoottorin vaihevirtojen katkontaan Alpha & Omega AO4882 -MOSFET-kanavatransistoreita (TMC5160-BOB BOM 2020).

Moottoriohjaimen käyttämä kellosignaali syötetään yhdestä Arduinon lähtönastasta. Tämä nasta on asetettu pulssinleveysmodulaatiomoodiin tuottamaan

kanttiaaltoa 50 % pulssisuhteella. Se muodostetaan mikrokontrollerin käyttämästä 84 MHz:n taajuisen sisäisen kellon signaalista jakoluvulla kuusi. Tällöin askelmoottoriohjaimen käyttämä kellotaajuus on 14 MHz.

2.4.1 StealthChop2-hakkuri

Trinamic TMC5160:ssa on Trinamic:n kehittämä oppiva StealthChop2-hakkuri. Se mahdollistaa hiljaisen ja tasaisen toiminnan matalissa nopeuksissa. StealthChop2 käyttää toiminnassaan pulssinleveysmodulaatiota, jolla saavutetaan mahdollisimman puhdas sinin aaltomuoto moottorin virralle ja täten tasainen liike. (Dwersteg 2019, 57).

StealthChop2 ei vaadi tarkkaa rekisteriarvojen säätöä ja alustusta, vaan se adaptoituu itse automaattisesti. Automaattisen adaptaation suoritus vaatii kaksi vaihetta: Pysähtymistilan ja adaptaatioajon. Ensimmäisessä vaaditaan 1572864 kellosyklin mittainen (noin 112 millisekuntia 14 MHz:n kellotaajuudella) pysähtymistila ajovirralla *IRUN.* Toinen vaihe, eli adaptaatioajo tarkoittaa kahdeksan fyysisen askeleen pituista askelmoottorin ajamista nopeudella, jolla moottorin johtimiin indusoituva vastasähkömotorinen voima on merkittävä (datakirjan mukaan yleensä nopeusalueella 60 – 300 kierrosta minuutissa). (Dwersteg 2019, 58).

Kyseinen adaptaatio toteutuu simulaattorissa automaattisen kalibroinnin ensimmäisessä vaiheessa. Vaikka *IHOLD*-arvoa ei nostettu *IRUN*-arvoon, vaikutti moottoriohjain adaptoituvan hyvin.

StealthChop2-hakkurille on olemassa myös muita alustusrekistereitä, joiden avulla virran aaltomuoto on mahdollista hienosäätää paremmaksi jo ennen automaattista adaptaatiosekvenssiä, mutta niiden asettamista ei koettu tarpeelliseksi.

2.4.2 SpreadCycle-hakkuri

Toinen simulaattorissa käytetty TMC5160:n ominaisuus on SpreadCycle-hakkurimoodi. Tämä hakkurimoodi tutkii jokaisen askeleen toteutunutta aaltomuotoa ja

pyrkii minimoimaan aaltomuodon ylimenosärön. StealthChop2 on hiljaisin ja tasaisin hakkurimoodi matalilla kierrosnopeuksilla, mutta nopeammissa pyörimisnopeuksissa se ei kykene kovin suuriin kiihtyvyyksiin. SpreadCyclen avulla saavutetaan korkeammilla kierroksilla parempi kiihtyvyys pienellä kuormalla, mutta silti säilytettyä tasainen liike. TMC5160:een on ohjelmoitavissa erikseen pyörimisnopeuden raja-arvo, jossa moottorinohjain vaihtaa hakkurimoodin StealthChop2:sta SpreadCycleen automaattisesti. (Dwersteg 2019, 68).

Verrattuna StealthChop2-hakkurimoodiin SpreadCycle vaatii enemmän alustusta ja kokeilua sopivien alustusarvojen löytämiseksi. Lopullisessa simulaattorissa ei toiminnassa havaittu juuri eroa StealthChop2:n ja SpreadCyclen välillä käytetyn raja-arvon (300 kierrosta minuutissa) yläpuolella, sillä askelmoottorin perässä oleva, muun muassa kiekoista ja hihnasta koostuva kuorma arveltiin olevan riittävän suuri StealthChop2:lle simulaattorissa ohjelmallisesti rajoitetulla maksimikiihtyvyydellä 1000 kierrosta minuutissa per sekunti. SpreadCycle päätettiin kuitenkin jättää käyttöön raja-arvon yläpuolella.

2.5 Automaattinen kalibrointi

Simulaattori suorittaa käynnistettäessä tai akryylistä valmistettua suojakantta laskettaessa automaattisen kalibrointisekvenssin, joka synkronoi askelmoottoriohjaimen sisäisen kulmatiedon ja kiekkojen asennon keskenään. Tämä tapahtuu kolmessa vaiheessa (kuva 4).

Ensimmäisessä vaiheessa etsitään sellainen simulaattorin kiekkojen asento, joka sijaitsee mahdollisimman lähellä vaihekiekon signaalin laskevaa reunaa. Tämän saavuttaakseen askelmoottoria pyöritetään myötäpäivään tai vastapäivään riippuen vaiheanturin signaalin tilasta kalibroinnin alkuhetkellä. Moottoria pyöritetään tasaisella nopeudella, kunnes vaihekiekon anturin ulostulojännite muuttuu ykkös-tilasta nollatilaan. Sitten askelmoottori pysäytetään.

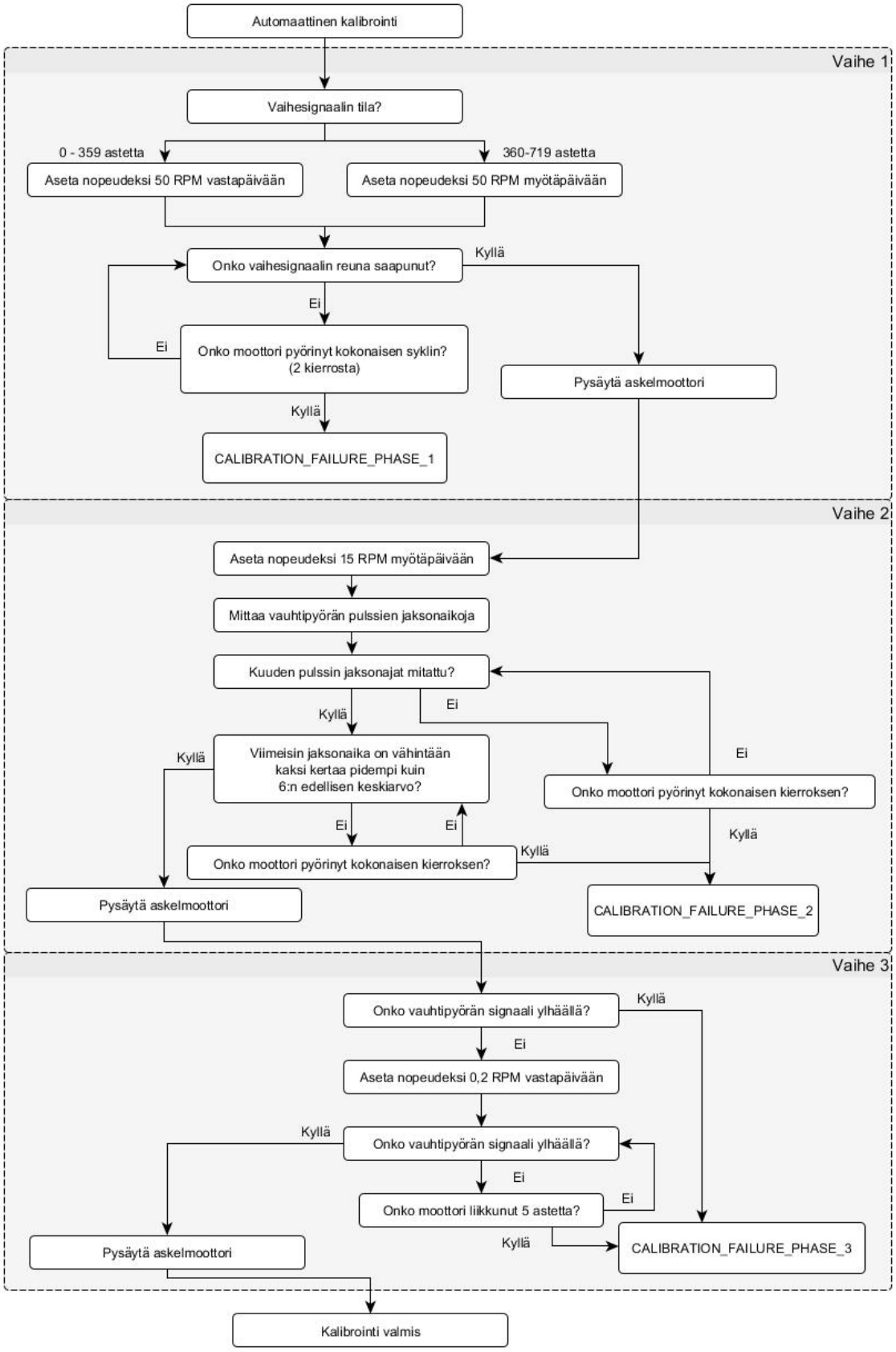
Kun vaihekiekon laskeva reuna on löytynyt, tiedetään kiekkojen nykyisen asennon olevan nyt likimain moottorisyklin positiossa 540 astetta. Position virhe riip-

puu vaihekiekon asennuksen tarkkuudesta kampiakselikiekkoon nähden. Voidaan päätellä, että seuraavaksi vastaan tuleva puuttuva hammas on askelmoottoria vastapäivään pyörittäessä moottorisyklin positiossa 360 astetta ja askelmoottoria myötäpäivään pyörittäessä moottorisyklin positiossa 0 astetta.

Toisessa vaiheessa simulaattori pyörittää askelmoottoria tasaisella nopeudella myötäpäivään ja tutkii samalla kampiakselin kiekon anturin tilamuutoksiin kuluvaa aikaa. Kun jaksonaika kaksinkertaistuu, voidaan simulaattorin olettaa olevan puuttuvan hampaan kohdalla, jolloin askelmoottori pysäytetään. Kalibrointiohjelma olettaa, että kiekossa on vain yksi puuttuva hammas.

Kolmannessa vaiheessa askelmoottoria pyöritetään miniminopeudella vastapäivään ja odotetaan kampiakselikiekon anturin tilamuutosta. Kun tilamuutos havaitaan, oletetaan kiekon olevan nyt positiossa nolla. Simulaattori pysäytetään ja simulaattorin askelmoottoriohjaimen sisäinen kulma nollataan.

Mikäli jokin ylläolevista vaiheista epäonnistuu, kun esimerkiksi puuttuvaa hammasta ei tunnisteta kokonaisella pyörähdyksellä, kalibrointisekvenssi pysäytetään ja aloitetaan uudestaan toisella anturilla. Mikäli tämäkin kalibrointi epäonnistuu, ilmoitetaan käyttäjälle, että simulaattorin positiomittaus ei ole käytettävissä. Simulaattorin nopeusmoodi on silti käytettävissä.



KUVA 4. Kalibrointisekvenssin vuokaavio yhdelle anturille

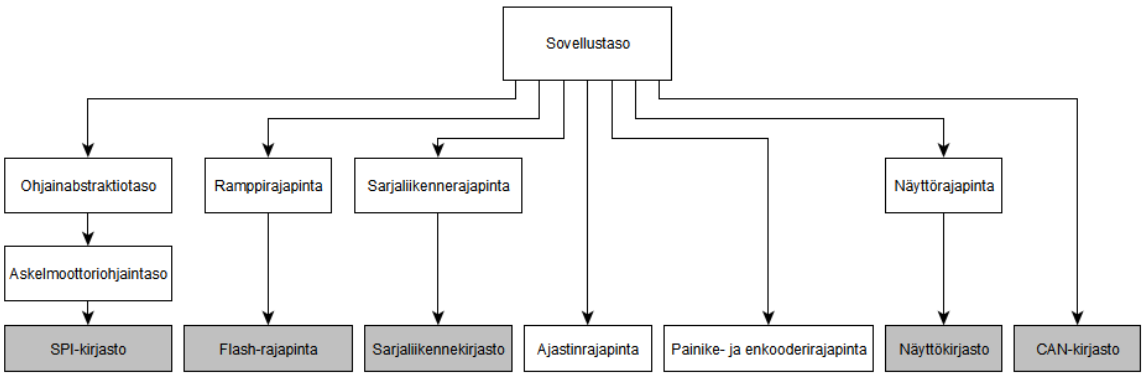
3 OHJELMISTON ARKKITEHTUURI

Vaikka Arduino-kehitysalustan kääntäjänä toimii C++-kääntäjä (GCC), niin ohjelmisto kirjoitettiin C-kielellä, sillä projektia ohjanneet, ohjelmakoodia katselmoineet henkilöt olivat siinä harjaantuneempia. Tällä nopeutettiin katselmointiprosessia ja varmistettiin parempi ohjelmiston laatu. Ohjelmalistauksessa on kuitenkin muutamia viittauksia C++-kieleen kohdissa, joissa käytetään Arduinon omia rajapintametodeita.

Kehityksessä sovellettiin valikoidusti MISRA-C:2012-suosituksen ohjeita, joilla edesautettiin helposti luettavan ohjelmalistauksen ja toimintavarman ohjelmiston kehitystä.

Ohjelmiston askelmoottoria ohjaava arkkitehtuuri koostuu kolmesta tasosta: askelmoottoriohjaintasosta, ohjainabstraktiotasosta ja sovellustasosta. Tässä opinnäytetyössä keskitytään askelmoottoriohjaintasoon tarkemmin myöhemmin.

Lisäksi ohjelmisto sisältää omat rajapintansa muun muassa näytön ohjaamiselle, painikkeiden lukemiselle, ramppimuistin hallinnalle ja sarjaliikennekomentojen tulkinnalle (kuva 5).



KUVA 5. Ohjelmiston arkkitehtuuri. Harmaat osat ovat käytettyjä valmiita kirjastokokonaisuuksia.

3.1 Ohjainabstraktiotaso

Ohjainabstraktiotaso on välitaso, joka luo mahdollisimman yksinkertaisen rajapinnan sovellustason käyttöön. Tason toiminnallisuudet olisi voinut sisällyttää ylempään tai alempaan tasoon, mutta tämä taso mahdollistaa saman ohjelmiston käyttämisen eri askelmoottoriohjainta käytettäessä. Se myös mahdollistaa nopeussignaalien simuloinnin kokonaan elektronisesti FPGA:lla tai mikrokontrollerilla mekaanisen simuloinnin sijaan vain ohjaintasoa vaihtamalla. Samalla ohjaintaso haluttiin pitää mahdollisimman yksinkertaisena, jotta yllämainitut muutokset olisivat mahdollisimman helppoja. Suurin osa tämän tason funktioista sisältää vain suoria kutsuja ohjaintason funktioihin parametrien tarkemmalla sovelluskohtaisella validoinnilla, mutta taso sisältää myös jonkin verran toiminnallisuutta, jonka ei koettu sopivan sovellus- eikä ohjaintasoon.

Esimerkkinä abstraktiotason toiminnallisuudesta on seuraava ohjaintason vapaasti kasvavan ja vähenevän positioarvon kääriminen nokka-akselin kulmaa vastaavaksi, 720 asteen pituisen moottorisyklin sisällä pyöriväksi positioarvoksi.

```
static uint32_t ToAbsolutePosition(const int32_t driverPosition_mdeg)
{
    int32_t absolutePosition_mdeg = abs(driverPosition_mdeg) % FULL_CYCLE_MDEG;
    if (driverPosition_mdeg < 0)
    {
        absolutePosition_mdeg = FULL_CYCLE_MDEG - absolutePosition_mdeg;

        if (absolutePosition_mdeg == FULL_CYCLE_MDEG)
        {
            absolutePosition_mdeg = 0;
        }
    }

    return absolutePosition_mdeg;
}
```

Siinä ohjainposition itseisarvolle tehdään jakojäännösoperaatio moottorisyklin pituuden kanssa. Mikäli alkuperäinen ohjainpositio on negatiivinen, täytyy juuri saatu absoluuttinen positioarvo vähentää simulaattorin syklistä. Tasan 720 000 milliastetta ei ole sallittu arvo, vaan on käytännössä sama positio kuin positio nolla.

3.2 Sovellustaso

Sovellustaso yhdistää kaikki alemmat rajapinnat ja luo simulaattorin käyttäjälle näkyvän käyttöliittymän. Suurin osa sovellustasosta koostuu käyttöliittymästä ja tulkittujen sarjaliikennekomentojen yhdistämisestä askelmoottoriohjaimen abstraktiotason funktioihin. Sovellustaso sisältää myös ohjelman pääsilmukan.

3.3 Muut kirjastot

Muita simulaattorin kannalta tärkeitä kirjastoja ovat muun muassa sarjaliikennekäyttöliittymän tulkintaa hallinoiva kirjasto, näyttökirjasto, näppäin- ja enkooderikirjasto paikallisten käskyjen tulkintaan, ramppipuskurikirjasto valmiiden nopeusnäytteiden tallennukseen mikrokontrollerin sisäiseen flash-muistiin ja ajastinkirjasto mikrokontrollerin ajastimien alustamista varten.

4 OHJAINKIRJASTON RAKENNE

Ohjainkirjasto koostuu staattisista sisäisistä funktioista ja kirjaston ulkopuolelle avoimista rajapintafunktioista. Jälkimmäiset ovat funktioita, jotka näkyvät rajapintaa käyttävälle ohjelmalle, ja sisäiset funktiot ovat kirjaston sisäisessä käytössä.

Kaikki funktiot, joissa jokin voi mennä pieleen, palauttavat paluuarvonaan tilaarvon, jolla kerrotaan, mikä mahdollisesti meni pieleen. Varsinaiset lasketut paluuarvot palautetaan funktion parametrina annettuun osoittimeen.

4.1 Ohjaimen alustustoimenpiteet

Ennen ohjaimen käyttämistä täytyy alustaa SPI-väylä ja TMC5160:n rekisterit halutulla tavalla. Nämä tehdään rajapintafunktiossa *TMCInit*. Mikäli mitä tahansa muuta rajapintafunktiota kutsutaan ennen alustusfunktion ajamista onnistuneesti, palauttavat ne virhetilan enumeraation *TMC_ERROR_NOT_INITIALIZED*, sillä ohjainkirjaston käyttäminen ei ole sallittua ennen alustusta.

Alustusfunktio alustaa SPI-väyläkommunikaatioon käytetyn kirjaston, askelmoottoriohjaimen muut ohjausulostulot, kaikki tarvittavat TMC5160:n rekisterit sopiviin alkuarvoihin sekä alustaa SPI-sarjaväylän kellona käytetyn mikrokontrollerin ulostulon PWM-tilaan tuottamaan kanttiaaltoa 7 MHz:n taajuudella. Kaikki luettavissa olevat alustetut rekisterit luetaan kirjoittamisen jälkeen, jotta varmistetaan SPI-kommunikaation luotettava toiminta.

Rekistereiden alustusarvoja optimoitiin datakirjan ja muun askelmoottoriohjaimen valmistajan tarjoaman dokumentaation pohjalta sekä kokeilemalla. Tasaiselle pyörimisliikkeelle tärkeimmät alustusarvot tallennetaan rekistereihin *PWMCONF*- ja *CHOPCONF*. Näillä hallitaan hakkurin ja sen pulssinleveysmodulaation ominaisuuksia. Niiden sopivat alkuarvot riippuvat käyttötarkoituksesta ja käytetystä moottorista.

4.1.1 Virtarajat

Otettaessa käyttöön uutta askelmoottoria täytyy virtarajarekisterit asettaa moottorille sopiviksi. TMC5160:n rekisterillä *GLOBAL SCALER* asetetaan moottorin virtarajan globaali skaalaus. Arvo on kahdeksanbittinen ja kuvastaa jakolukua moottoriohjaimen maksimivirranantokyvylle. Datakirjan mukaan skaalausjakajat 0-31 eivät ole sallittuja (Dwersteg 2019, 36).

IRUN-rekisterillä asetetaan normaali ajovirta, eli virtaraja kun moottoria liikutetaan normaalisti. Simulaattorin normaalissa toiminnassa tämä arvo on asetettu maksimiinsa, eli arvoon 31, sillä moottorille sopiva virtaraja on käytännössä asetettu globaalilla skaalauksella. Kyseistä rekisteriarvoa kuitenkin muutetaan erityistilanteissa.

IHOLD-rekisteri rajoittaa virran määrää tilanteessa, jossa moottoria pidetään paikoillaan. Datalehti suosittaa tähän arvoa, joka on noin 70 prosenttia IRUN-rekisterin arvosta, eli tässä tapauksessa päädyttiin arvoon 21 (Dwersteg 2019, 111).

Kun käytössä oli TMC5160-BOB-kehitysalusta 2,8 ampeerin virransyöttökyvyllä ja normaalissa toiminnassa on asetettu jakaja-arvo 150, on tällöin moottorille rajoitettu virta ajossa likimain kaavan 1 mukainen (Dwersteg 2019, 74). Kaavassa *CS* on rekisterin *IRUN*- tai *IHOLD*-arvo riippuen siitä, onko moottori ajossa vai paikallaan. V_{FS} on virtatunnistuksen kynnysjännite, eli tässä tapauksessa noin 325 millivolttia (Dwersteg 2019, 123). R_{SENSE} on virtatunnistusvastuksen resistanssi, joka TMC5160-BOB-kytkennässä on 75 milliohmia (TMC5160-BOB Datasheet 2020, 2).

$$I_{RMS} = \frac{GLOBAL\_SCALER}{256} \cdot \frac{CS + 1}{32} \cdot \frac{V_{FS}}{R_{SENSE}} \cdot \frac{1}{\sqrt{2}} \approx 1{,}8\ \text{A} \quad (1)$$

Virtarajoilla on huomattava vaikutus moottorin tasaiseen liikkeeseen, ja sopivien arvojen löytäminen vaati kokeilua. Suurin vaikutus on globaalilla virtaskaalauksella *GLOBAL SCALER*.

4.1.2 Hakkurirekisterit

Simulaattorissa haluttiin käyttää pienemmille nopeuksille StealthChop2-hakkurimoodia ja suuremmille nopeuksille SpreadCycle-hakkuria. Tämän saavuttamiseksi piti muutamia rekistereitä asettaa sopiviin alustusarvoihin.

Pääkonfiguraatiorekisteri *GCONF* sisältää monia askelmoottoriohjaimen yleiseen käyttöön liittyviä asetuksia, mutta käytettäessä SPI-väylällä ja ilman TMC5160:n erillisiä diagnostiikkaulostuloja ainoastaan asetus *en_pwm_mode* oli tarpeen kirjoittaa arvoon 1. Kyseinen asetus ottaa StealthChop-hakkurimoodin käyttöön.

StealthChop2-hakkurin asetukset kirjoitetaan *PWMCONF*-rekisteriin. Rekisterin *pwm_ofs*- ja *pwm_grad*-biteillä asetetaan hakkurin aaltomuotoasetukset, mutta niillä ei ole paljoa merkitystä sikäli, että sitä käytetään vain ennen StealthChop2:n automaattista kalibrointia ja sen aikana. *Pwm_ofs*-arvoksi asetettiin 200, joka rajoittaa virran aaltomuodon amplitudin melko pieneksi ennen kalibrointia. *PWM_GRAD* asetettiin alustuksessa arvoon 1, joten ennen kalibrointia aaltomuodon amplitudia ei kompensoida nopeuden mukaan. Arvojen muutokset eivät muuttaneet askelmoottorin käytöstä, joten näiden arvojen etsimiseen ei käytetty juuri aikaa. Biteillä *pwm_autoscale* ja *pwm_autograd* nämä arvot asetetaan automaattisesti säädettäväksi automaattisen kalibroinnin yhteydessä. *PWMCONF*-biteillä *pwm_freq* valittiin PWM-taajuudeksi 41 kHz ($2/683 \cdot f_{clk}$), joka on datakirjan taulukon 7.1 mukaan sopiva käyttötaajuus. (Dwersteg 2019, 60).

CHOPCONF-rekisteri sisältää asetuksia, jotka vaikuttavat sekä StealthChop:n että SpreadCycle:n toimintaan (Dwersteg 2019, 51). Tämän rekisterin arvoilla *toff* (alhaallaoloaika), *hstrt* (amplitudin hystereesin yläraja) ja *hend* (amplitudin hystereesin alaraja) asetetaan SpreadCyclen käyttämän siniaallon asetukset. Näiden asetusten löytämiseen käytettiin datakirjan ohjeita ja kokeiltiin eri arvoja, kunnes toiminta korkeammilla nopeuksilla (yli 300 kierrosta minuutissa) oli tasaista ja mahdollisimman äänetöntä (Dwersteg 2019, 69-71). Vastaavasti *tpfd*-arvo (heikentymisaika) päätettiin tutkimalla moottorin käyttäytymistä keskinopeuksilla eri arvoilla. Arvoksi valittiin sellainen, jolla simulaattorin ulostulojen värähtelyt ovat

oskilloskoopilla ja silmämääräisesti tutkittuna kaikista pienimmät. *TBL*-arvo (hakkurin tyhjä aika katkontatapahtuman jälkeen) valittiin vertailemalla hieman eri arvoja. Suurta eroa eri arvojen välillä ei havaittu, joten valittiin datakirjan suosittama arvo 2 (Dwersteg 2019, 51). Rekisterin bitillä *chm* valittiin käytettäväksi SpreadCycle-hakkuri (asetettiin nollaksi) ja *vhighfs*-bitti asetettiin nollaksi. Jälkimmäisen bitin avulla voitaisiin moottoria tarvittessa käyttää täysaskelilla korkeilla kierroksilla *VHIGH*-rekisteriin asetetun nopeuden ylittyessä. Rekisterin biteillä *mres* asetettiin käytettäväksi mikroaskellusmääräksi 256 (arvo 0). *CHOPCONF*-rekisterin määrittely ohjelmalistauksessa kommentteineen:

```
// Chopper configuration
// TPFD = 9 TOFF = 3, HSTRT = 0, HEND = 9, TBL = 2, CHM = 0 (spreadcycle),
// vhighfs = 0
#define CHOPCONF  0x00910583
```

Lopuksi asetettiin *tpwmthrs*-rekisteriin nopeusraja-arvo, jossa StealthCycle2-hakkurimoodista vaihdetaan käyttämään SpreadCycle-hakkuria. Valittiin arvo, joka vastaa nopeutta 300 kierrosta minuutissa.

4.2 Ohjainkirjaston sisäiset funktiot

Ohjainkirjaston useimmin käytetyt sisäiset funktiot ovat TMC5160:n rekistereiden luenta- ja kirjoitusfunktiot.

TMC5160:n kanssa kommunikointiin käytetty SPI-kehys koostuu 8-bittisestä otsikkotiedosta ja 32-bittisestä datasta. Otsikossa lähetetään kirjoitettaessa seitsemässä ensimmäisessä bitissä halutun rekisterin arvo. Eniten merkitsevä otsikkosanan bitti määrittää, halutaanko osoitteen rekisteriä lukea vai kirjoittaa. Esimerkiksi jos halutaan askelmoottorin nopeudeksi (rekisteriin *VACTUAL* osoitteessa 0x22) asettaa heksadesimaaliarvo 0xFF, lähetetään ohjaimelle taulukon 1 mukainen SPI-kehys. (Dwersteg 2019, 23 – 25.)

TAULUKKO 1. Esimerkki SPI-kehyksestä askelmoottoriohjaimelle.

	Otsikko		**Data**
	RW-bitti	Osoite	
Binäärinen	1	0100010	00000000 00000000 00000000 11111111
Heksa-desimaalinen	0xA2		0x000000FF

Mikäli ohjaimen rekisteriä luetaan, kehyksen dataosuus jää ohjaimessa huomiotta. Ohjainkirjasto kirjoittaa tällöin datan sisällöksi pelkkiä nollabittejä.

Koska SPI-väylässä sekä mikrokontrolleri että TMC5160 lähettävät kehyksen yhtä aikaa mikrokontrollerin syöttämän SPI-kellon mukaisesti, on lukufunktio käytännössä kirjoitusfunktio kaksi kertaa peräkkäin. Kun mikrokontrolleri kirjoittaa SPI-väylän avulla pyynnön halutun arvon lukemisesta, palauttaa askelmoottoriohjain sen seuraavalla mikrokontrollerin SPI-kirjoitushetkellä. Käytännössä lukufunktion jälkimmäisellä kerralla mikrokontrolleri lähettää tyhjän kehyksen, jolloin samaan aikaan askelmoottoriohjain palauttaa edellisellä kirjoituskerralla pyydetyn rekisteriarvon.

TMC5160 lähettää jokaisessa SPI-kehyksen otsikkotavussa tilabittiensä arvot (Dwersteg 2019, 24). Ohjainkirjasto käyttää tilabiteistä kolmea: bittiä neljä, joka ilmaisee saavutetun tavoitenopeuden, bittiä viisi, joka ilmaisee saavutetun tavoiteposition ja bittiä yksi, joka ilmaisee havaitun rautatason ongelman, jonka voi lukea tarkemmin *GSTAT*-rekisterin virhelippujen arvoista.

4.2.1 Rekisteriarvojen konversiot

Ohjainkirjasto muuttaa TMC5160:n nopeusrekisterin *VACTUAL*-arvon millikierroksiksi minuutissa, kiihtyvyysrekisterit *A*, *A1*, *D*, *D1* millikierroksiksi minuutissa per sekunti ja positiorekisterit *XACTUAL* ja *XTARGET* milliasteiksi sisäisellä laskennalla. Laskenta perustuu TMC5160:n datalehden ilmoittamiin laskukaavoihin (Dwersteg 2019, 81). Koska kyseisiä funktioita käytetään usein, haluttiin niiden suorituksesta mahdollisimman nopea. Tällöin täytyi välttää liukulukulaskentaa,

sillä simulaattorissa käytetyn mikrokontrollerin Cortex M3 -ydin ei sisällä matematiikkasuoritinta (FPU) (Arm Cortex-M Series Processors -tuotesivu 2020). Kaavojen mukaan laskettaessa muunnettava luku käy muunnoksen laskutoimitusten välituloksissa todella suurissa ja todella pienissä arvoissa. Tämä aiheutti ongelman, sillä kokonaisluvuilla laskettaessa pyöristysvirheet kasvoivat valtaviksi.

Ratkaisuksi valittiin laskenta 64-bittisenä lukuna siten, että laskutoimitusten jaettavia skaalattiin suuremmiksi hyväksi koetuilla kertoimilla, jolloin pienien kokonaislukuvälitulosten katkaisusta johtuvat virheet minimoituivat. Skaalauskerroin jaettiin pois, kun tulos oli kertolaskun myötä kasvanut riittävän suureksi. Kyseinen ratkaisu vaati myös laskutoimitusten järjestelemistä skaalaukselle suotuisaan järjestykseen. Eri yksikkömuunnoksissa kertoimet ja laskut ovat hieman erilaisia. Esimerkkinä positiorekisteriarvon muunnos milliasteiksi:

```
static int32_t RegToMilliDeg(const int32_t position_reg)
{
    /*
    Original formulas for this conversion can be found from TMC5160 datasheet
    section 12.1 - Real World Unit Conversion.

    Register position value is in microsteps, so calculation is done by
    dividing full rotation with microsteps in rotation. To avoid underflow
    while calculating small values, numerator in multiplier-constant calculation
    (360000 mdegrees) is multiplied by ANGLE_CONVERSION_SCALER and then divided
    by it at the end of the conversion.

    Some constants need to be casted to 64-bit to avoid overflow.
    */
    const uint32_t multiplier =
        (((uint64_t)FULL_REV_MDEG * ANGLE_CONVERSION_SCALER) / (FSC_FS_PER_REV * USC_US_PER_FS));
    int64_t conversion_mdeg = (multiplier * (int64_t)position_reg) / ANGLE_CONVERSION_SCALER;

    return (int32_t)conversion_mdeg;
}
```

Optimoitujen laskukaavojen virhe jäi nopeuden laskennassa noin ± 1 millikierrokseen minuutissa, kiihtyvyyden virhe noin ± 5 millikierrokseen minuutissa per sekunti ja position virhe noin ± 10 milliasteeseen. Nopeuden ja kiihtyvyyden muunnos riippuu moottoriohjaimen kellotaajuudesta, joten vakaampi kellosignaali takaa tarkemmat tulokset.

4.3 Ohjainkirjaston rajapintafunktiot

Suurin osa ohjainkirjaston rajapintafunktioista koostuu erilaisista nopeus- ja kiihtyvyysrekisterien kirjoitusfunktioista, jotka muuntavat annetun reaalimaailman yksiköissä annetun arvon rekisteriarvoksi ja kirjoittavat sen askelmoottoriohjaimelle, ja lukufunktioista, jotka lukevat halutun arvon askelmoottoriohjaimelta ja muuntavat sen reaalimaailman yksiköiksi. Esimerkkinä seuraava moottorin pyörimisnopeuden moottoriohjaimelle asettavan rajapintafunktion esittely:

```
/******************************************************************************/
/*! \brief Sets max velocity register (VMAX)

    This register is used in constant speed driving. Velocity is varied by
    changing value of this register. This function does not check if driver
    really sets that value.

    Upper limit of velocity value is maximum register
    value of the VMAX. So in mrpm it depends to clock frequency used in
    TMC5160. In example: @14 Mhz clock to TMC5160 maximum velocity is
    8202623 mrpm and therefore minimum value is - 8202623.

    Values between -10 mrpm and 10 mprm are written as 0 because they are
    not allowed in the driver, except 0 = standstill.

    \param[in] n_velocity_mrpm Wanted velocity in mprm

    \return Returns driver function status code.
            TMC_SUCCESS - new velocity written to driver successfully.
            TMC_ERR_NOT_INITALIZED - driver not initialized
            TMC_ERR_VALUE_TOO_BIG - velocity_mrpm value is too big
            TMC_ERR_DRV_DISABLED - Driver is disabled.
            Errors passed from TMCGetDirection -function
            Error codes passed from WriteData -function
******************************************************************************/
TMCStatus TMCSetMaxVelocity(int32_t n_Velocity_mrpm);
```

4.3.1 Toimintamoodit

TMC5160 sisältää neljä eri ohjaustilaa, jota vaihdetaan muuttamalla rampigeneraattorin ohjausrekisterin kohtaa *RAMPMODE*. Moodi 0 on positiomoodi, jossa ohjain ajaa asetetuilla kiihtyvyys-, hidastuvuus- ja maksiminopeusarvoilla moottorin haluttuun positiorekisterin *XTARGET* arvoon. Moodit 1 ja 2 ovat nopeusmoodeja. Moodi 1 ajaa moottoria positiivisella *VMAX*-rekisterin nopeudella

(simulaattorissa pyörintäsuunta myötäpäivään) ja moodi 2 ajaa moottoria negatiivisella rekisterin *VMAX*-nopeudella (simulaattorissa pyörintäsuunta vastapäivään). Molemmat nopeusmoodit käyttävät *AMAX*-rekisterin kiihtyvyyttä sekä kiihdyttäessä että hidastettaessa. Moodi 3 on pitomoodi, joka pitää askelmoottorin nopeuden vakiona. Kyseistä moodia ei käytetty tässä sovelluksessa. (Dwersteg. 2019, 40).

Nämä moodit tiivistettiin ohjainrajapinnassa kahteen eri rajapintamoodiin: nopeusmoodiin, jossa askelmoottoria ajetaan myötä- tai vastapäivään riippuen asetetun nopeuden etumerkistä, sekä positiomoodiin, jolla askelmoottoria voidaan ajaa haluttuun positioon. Rajapinnan positiomoodi on käytännössä sama kuin moottoriohjaimen moodi 1.

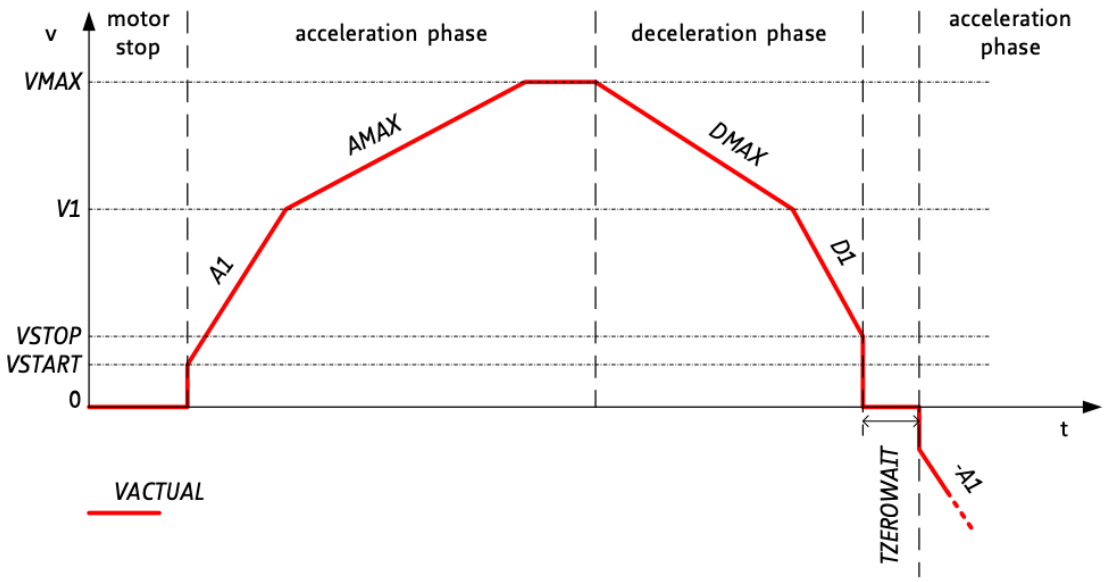
4.3.2 Positiomoodin käyttö

Rajapintaan suunniteltiin funktio *TMCResetPosition*, joka nollaa TMC5160:n sisäisen, nykyisen moottorin position sisältävän *XACTUAL*-rekisterin. Tämä nollaus suoritetaan esimerkiksi kalibrointisekvenssissä, mutta myös muutamissa muissa erikoistapauksissa, joista tarkemmin jäljempänä.

Funktion *TMCSetPosition* avulla ajetaan askelmoottori haluttuun positioon. Parametrina funktiolle annetaan kohdepositio, joka on etäisyys nollatusta positiosta milliasteina. Moottoriohjaimella ja ohjaintasolla moottorin positio pyörii vapaasti 32-bittisessä etumerkillisessä rekisterissä ja ylempi ohjaimen abstraktiotaso käärii etäisyysarvon pyörimään 720 asteen moottorisyklin sisällä.

Positiomoodi toimii TMC5160:lla kahdella eri kiihtyvyydellä ja kahdella eri hidastuvuudella (kuva 6). Kun ohjainkirjasto muuttaa uuden milliastearvon rekisteriarvoksi ja asettaa sen *XTARGET*-rekisteriin askelmoottoriohjaimen ollessa moodissa 1, ajaa TMC5160 askelmoottoria kohti uutta positiota kiihtyvyydellä, joka on asetettu rekisteriin *A1*. Kun rekisteriin *V1* asetettu nopeus ylitetään, vaihtuu kiihtyvyys *AMAX*-rekisteriin asetettuun arvoon ja nopeus kiihtyy, kunnes rekisteriin *VMAX* kirjoitettu nopeus saavutetaan. Kun askelmoottoriohjain hidastaa halutun position lähestyessä, se käyttää ensin rekisteriin *DMAX* asetettua hidastuvuutta

ja kun *VMAX*-nopeus on alitettu, vaihdetaan hidastuvuus arvoon, joka on kirjoitettu rekisteriin *D1*. (Dwersteg 2019, 82).



KUVA 6. Positiomoodin kiihtyvyydet ja hidastuvuudet. (Dwersteg 2019, 82)

Simulaattorin ohjaimen abstraktiotaso kirjoittaa rekistereihin *A1*, *V1* ja *D1* arvon nolla alustuksen yhteydessä, jolla kyseinen kaksiasteinen kiihdytys otetaan pois käytöstä ja käytetään vain rekistereiden *AMAX*-, *DMAX*- ja *VMAX*-arvoja. Ohjainrajapintaan suunniteltiin funktiot kyseisen toiminnallisuuden käyttämiseen sitä haluttaessa. Rajapinnassa nämä arvot asetetaan funktioilla *TMCSetPositionModeStartingAcceleration* (*A1*-rekisteri), *TMCSetPositionModeStartingDeceleration* (D1-rekisteri), *TMCSetPositionModeStartingVelocity* (*V1*-rekisteri), *TMCSetMaxAcceleration* (*AMAX*-rekisteri), *TMCSetMaxDeceleration* (*DMAX*-rekisteri) ja *TMCSetMaxVelocity* (*VMAX*-rekisteri.)

4.3.3 Nopeusmoodin käyttö

Nopeusmoodissa askelmoottoria pyöritetään vakionopeudella. Tavoitenopeus asetetaan kirjoittamalla nopeusarvo rekisteriin *VMAX*. Käytetty kiihtyvyys ja hidastuvuus asetetaan rekistereihin *AMAX* ja *DMAX*. Simulaattorissa ei erikseen aseteta hidastuvuutta, vaan samat kiihtyvyysarvot kirjoitetaan sekä *AMAX*- että *DMAX*-rekisteriin.

Ohjaimen asettamiseen eri moodeihin suunniteltiin rajapintafunktio *TMCSetMode*, jolle parametrina annetaan esimerkiksi nopeusmoodin tapauksessa enumeraatio *TMC_VELOCITY*. Nopeuden säätelyyn suunniteltiin rajapintafunktio *TMCSetMaxVelocity*, joka muuntaa milliasteita minuutissa -arvon rekisteriarvoksi ja kirjoittaa sen *VMAX*-rekisteriin.

Moottoriohjaimen datakirja suosittaa välttämään nopeuksia alle *VMAX* rekisteriarvon 12, joka muunnettuna ja ylöspäin pyöristettynä on noin 10 milliastetta sekunnissa. Tämä johtuu siitä, että todella hitaissa nopeuksissa moottorin pyörimisliikkeestä syntyviin ilmavirtauksiin perustuva moottorin jäähdytys ei toimi, ja vaarana on, että moottori menee epäkuntoon ylikuumenemisen takia. Ohjainrajapinnan nopeudenasetusfunktioon lisättiin tarkistus, jolla rajoitetaan nopeuden asetus ala-arvoon 10 milliastetta minuutissa. Pienemmän nopeuden asetus kirjoittaa moottoriohjaimelle moottorin pysäyttävän nopeusarvon 0.

4.3.4 Vapaa pyöritys

Vapaa pyöritys (eng. *freewheeling*) on tila, jossa moottorinohjain katkaisee kaikki askelmoottorin positiota ylläpitävien kanavatransistorien virrat, jolloin simulaattorin kiekkoja pystytään liikuttamaan käsin. Käytännössä tämä tila asettaa askelmoottoriohjaimen virtarajarekisterit arvoon nolla.

5 OHJAINKIRJASTON ERIKOISTOIMINNOT

Simulaattoria kehitettäessä havaittiin useita erikoistilanteita, jotka vaativat ohjaimelta erikoistoimenpiteitä. Erikoistoiminnot ovat sellaisia funktioita, joita käyttäjän täytyy suorittaa periodisesti vaikkapa kymmenen millisekunnin välein. Näitä erikoistoimintoja varten luotiin ohjainabstraktiotasolle funktio, joka suorittaa kaiken ohjaimella tarvittavan periodisen käsittelyn.

5.1 XACTUAL-rekisterin ylivuoto-ongelman korjaus

Nopeasti havaittiin ongelma, jossa synkronointi kampiakselikiekon ja askelmoottoriohjaimen sisäisen position välillä katoaa simulaattoria ajettaessa niin kauan, että sisäinen *XACTUAL*-rekisteri pääsee ylivuotamaan.

Tämä korjattiin lisäämällä ohjaimeen suojamekanismifunktio *TMCIsPositionLargerThanRevertLimit*, joka suoritetaan periodisesti kymmenen millisekunnin välein:

```
bool TMCIsPositionLargerThanRevertLimit(uint64_t positionToCheck_mdeg)
{
    if (abs((int32_t)MilliDegToReg(positionToCheck_mdeg)) > XACTUAL_REVERT_LIMIT)
    {
        return true;
    }
    else
    {
        return false;
    }
}
```

Tämä suojamekanismi tarkistaa onko askelmoottoriohjaimen sisäinen tila ylittänyt lähellä ylivuodon rajaa olevan arvon. Jos positio on ylitetty, palauttaa funktio arvon tosi. Tällöin sitä kutsunut ylemmän tason funktio kutsuu ohjaimen rajapintafunktiota *TMCMinimizePositionRegisterValue*, joka asettaa *XACTUAL*-rekisterin arvon lähimpänä nollaa olevaan abstraktiotason ilmoittamaan käärittyyn positioarvoon.

5.2 Resonanssin aiheuttaman värähtelyn korjaus

Kun askelmoottoria ajettiin tietyllä nopeusalueella, havaittiin, että se värähtelee huomattavasti. Tämä värähtely johtui vastasähkömotorisen voiman aiheuttamasta resonanssista, joka riippuu askelmoottorin vaiheille syötettävän signaalin taajuudesta ja virrasta. Värähtely oli niin voimakasta, että se oli selvästi nähtävissä simulaattorin kampiakselikiekosta paljaalla silmällä. Apua kysyttiin askelmoottoriohjaimen valmistajalta Trinamicilta, josta neuvottiin optimoimaan askelmoottoriohjaimen sisäistä mikroaskelaaltomuototaulukkoa. Mikroaskelaaltomuototaulukon avulla voidaan kanavatransistorien jänniteohjauksen aaltomuotoa korjata, koska askelmoottorin aiheuttaman kuorman epälineaarisuudet aiheuttavat aaltomuodon vääristymistä. Tämän takia mikroaskeleet eivät jakaudu tasaisesti yhden moottorin fyysisen askeleen sisälle ja aiheuttavat resonanssin tietyllä taajuusalueella. Trinamicin ohjeiden mukainen mikroaskelaaltomuototaulukon optimointi ei juuri pienentänyt värähtelyä (Dwersteg, B. & Dwersteg, S. 2016).

Lopullinen korjaus oli muuttaa TMC5160:n käyttämää virtarajaa resonanssialueella sen verran, ettei resonanssia tapahdu. Ohjainrajapintaan lisättiin funktio *TMCResonanceCompensation*, jota täytyy suorittaa periodisesti. Funktiolle annetaan parametrina resonanssinopeuden ylä- ja alarajataajuus. Funktiossa tutkitaan sitä, onko nykyinen nopeus annetulla alueella. Jos on, asetetaan virtarajaksi *IRUN* sellainen, jolla resonanssia ei esiinny kyseisellä nopeusalueella. Kun funktiota kutsutaan uudelleen siten, että kyseiseltä alueelta ollaan poistuttu, palautetaan virtaraja *IRUN* ennalleen.

Nämä virtarajan muutokset tehdään kuitenkin vasta, kun askelmoottoria ohjaavien siniaaltojen amplitudit ovat nolla, eli kun moottoria ei pidätellä mikroaskelluksen avulla fyysisten askelien välissä. Tieto siniaaltojen amplitudista nolla saatiin askelmoottoriohjaimen mikroaskelluslaskurirekisterin *MSCNT* arvosta 0 (Dwersteg 2019, 47), jolloin askelmoottori on fyysisen askeleen kohdalla.

Tämän avulla saatiin minimoitua virtarajan muuttamisesta aiheutunut nykäys. Virtarajan muuttaminen kesken ajon aiheuttaa silti askelmoottorissa pienen nykäyksen, joka on kuultavissa naksahduksena, mutta ei nähtävissä simulaattorin ulostulon aaltomuotoa tutkittaessa oskilloskoopin avulla.

6 POHDINTA

Simulaattori on osoittautunut toimivaksi, vaikka kyseessä olikin soveltuvuusselvitys. Se on ollut aktiivisessa käytössä nelitahtimoottorin tuotekehitysprojektissa ja käyttäjät ovat kokeneet sen helppokäyttöisemmäksi ja luotettavammaksi kuin aiemmat ei-mekaaniset simulaattorit, varsinkin tutkittaessa moottorin ohjausjärjestelmän toimintaa lähellä nollanopeutta.

Ohjelmistoon on tehty muutamia korjauksia ja parannuksia. Muun muassa CAN-väylätuki lisättiin talvella 2019, kun simulaattorin sisäistä positiotietoa piti lähettää toiseen järjestelmään reaaliaikaisesti. Tiedossa on myös muutamia ohjelmistovikoja, joiden esiintymistä ei saatu toistumaan.

Laitteen mekaaninen kestävyys jää vielä nähtäväksi, mutta toistaiseksi ainoa tarvittava mekaanisten osien huoltotoimenpide on ollut kampi- ja nokka-akselien laakerien vaihtaminen laadukkaampiin vanhojen heikompien väljistyttyä.

Ohjainrajapinta on osoittautunut toimivaksi ja tehokkaaksi vaikka välissä onkin useampi ohjelmistokerros. Toki optimoinnin ja kehityksen varaa olisi. Nykyisellään moottoriohjaimelta arvon lukemiseen tarvitaan kaksi SPI-kirjoitusoperaatiota, kun joissain tapauksissa riittäisi, että vastaus luetaan vasta, kun suoritetaan seuraava kehyksen kirjoitus, jolloin samaan aikaan voitaisiin lukea edellisessä operaatiossa moottoriohjaimelta kysytty arvo. Nykyisessä toteutuksessa ei ole tiedossa, millä ajanhetkellä seuraava SPI-väylään kirjoitus tullaan suorittamaan, joten vastauksen saaminen ja täten kysytyn arvon vastaanottaminen ei ole ennustettavissa. Ratkaisuna tähän olisi voinut olla esimerkiksi ajastimeen perustuva periodinen SPI-kommunikaatio moottoriohjaimen kanssa ja viestipuskuri.

Toinen pohdinnan arvoinen seikka on se, että simulaattorin ohjelmiston olisi voinut alun perin varustaa jollain reaaliaikakäyttöjärjestelmällä. Vaikka ohjelmisto on monimutkainen, olisi prosessorin kuormitusaste silti jäänyt luultavasti melko matalaksi, sillä laskentatehoa on käytettävissä käyttötarkoitukseen nähden paljon. Käyttöjärjestelmä olisi luultavasti parantanut ohjelmiston rakennetta jonkin verran, ja tehnyt ohjelmistosta deterministisemmän, kun eri tehtäville pystyttäisiin

antamaan eri prioriteetteja. Tosin käyttöjärjestelmä olisi luultavasti tuonut mukanaan aivan uusia ongelmia ja kehitysaika olisi pidentynyt huomattavasti. Nykyinen toteutus nojaa prosessorin suorituskykyyn suorittaa kaikki tarvittavat tehtävät halutussa ajassa silmukan sisällä tietyin ajastimen aikavälein. Kehitysajan kannalta kevyempi parannus olisi varmistaa deterministisyyttä ajonaikaisesti tutkimalla, että esimerkiksi yhden millisekunnin välein suoritettavat toiminnot eivät ole viivästyneet liikaa muiden tehtävien suorituksen aikana. Järjestelmä ei voisi tehdä asialle mitään, mutta kehittäjä tietäisi milloin jokin funktio kaipaa optimointia, jotta järjestelmä saadaan pidettyä deterministisenä, vaikka ohjelmistoa muutetaan tai lisätään toimintoja.

Suunnittelun jälkeen on käynyt myös ilmi, että datakirjaa olisi voinut tulkita tarkemmin. Muun muassa hakkurin automaattisen kalibroinnin osalta prosessia ei tehdä täysin datakirjan ohjeita noudattaen. Kalibrointi kuitenkin toimii, mutta ehkä vain sattumalta, ja deterministisemmän toiminnan takaamiseksi olisikin suotavaa, että kyseinen kalibrointi tehtäisiin joka kerta samalla tavalla datakirjan ohjeiden mukaisesti.

Erityisen mainittavaa on MISRA-C:2012-ohjeiden käytön ja tarkan katselmointiprosessin hyödyt kehityksessä. Ohjelmalistaus on helposti luettavaa vielä pitkän ajan jälkeenkin ja hyvin johdonmukaista. Suurin osa tarvittavasta informaatiosta on sisällytettynä ohjelmalistaukseen ja luultavasti kuka tahansa C-ohjelmointikielen osaava kehittäjä pystyy omaksumaan ohjelmiston toiminnan kattavasti ilman muuta dokumentaatiota. Poikkeuksena ovat esimerkiksi moottorinohjaimen rekisterimuunnokset reaalimaailman yksiköiksi, joiden täysi ymmärtäminen voi vaatia hieman TMC5160:n datakirjan selaamista. Katselmointiprosessissa kokeneet C-ohjelmointikielellä yleensä työskentelevät kehittäjät kävivät ohjelmalistauksen läpi ja kyseinen ohjelmalistausosa liitettiin muuhun ohjelmistoon vasta sitten, kun katselmoijat olivat sen hyväksyneet.

Voitaisiin sanoa, että se mikä näyttää hyvältä, myös toimii hyvin. Siisti ja johdonmukainen ohjelmalistaus siis takaa myös virheiden näkyvyyden, eivätkä ne jää piiloon monimutkaisen ja sotkuisen koodin taakse. Tällöin virheet pystyttiin havaitsemaan jo aikaisessa kehitys- ja katselmointivaiheessa.

LÄHTEET

Arm Limited. Arm Cortex-M Series Processors. Tuotesivu. Luettu 6.4.2020. https://developer.arm.com/ip-products/processors/cortex-m

Dwersteg, B. 2019. TMC5160 / TMC5160A Datasheet. Trinamic Motion Control GmbH & Co. KG. Versio 1.13. https://www.trinamic.com/fileadmin/assets/Products/ICs_Documents/TMC5160A_Datasheet_Rev1.13.pdf

Dwersteg, B. & Dwersteg, S. 2016. Programmable Adaptive Microstep Table. Trinamic Motion Control GmbH & Co. KG. Versio 1.00d. https://www.trinamic.com/fileadmin/assets/Support/Appnotes/AN026-Adaptive_Microsteptable.pdf

Trinamic Motion Control GmbH & Co. KG. 2017. TMC5160-BOB BOM V.1.2. https://www.trinamic.com/fileadmin/assets/Products/Eval_Documents/BOM_TMC5160_BOB_V1.2_modified.xls

Trinamic Motion Control GmbH & Co. KG. 2017. TMC5160-EVAL BOM V.3.1. https://www.trinamic.com/fileadmin/assets/Products/Eval_Documents/TMC5160-EVAL_V31-BOM.xls

Trinamic Motion Control GmbG & Co. KG. 2019. TMC5160-BOB Datasheet V1.10. https://www.trinamic.com/fileadmin/assets/Products/ICs_Documents/TMC5160-BOB_datasheet_Rev1.1.pdf

Trinamic Motion Control GmbH & Co. KG. TMC5160-EVAL. Tuotesivu. Luettu 16.1.2020. https://www.trinamic.com/support/eval-kits/details/tmc5160-eval/

Trinamic Motion Control GmbH & Co. KG. TMC5160-BOB. Tuotesivu. Luettu 16.1.2020. https://www.trinamic.com/support/eval-kits/details/tmc5160-bob/