TALLINNA TEHNIKAÜLIKOOL
MEHAANIKATEADUSKOND

Mehhatroonikainstituut

Mehhatroonikasüsteemide õppetool

MHK40LT

Madis Taimre

TARKVARA 3D PRINTERIT JUHTIVALE MIKROKONTROLLERILE

Bakalaureusetöö

Autor taotleb
tehnikateaduse bakalaureuse
akadeemilist kraadi

Tallinn
2014

AUTORIDEKLARATSIOON

Deklareerin, et käesolev lõputöö on minu iseseisva töö tulemus.
Esitatud materjalide põhjal ei ole varem akadeemilist kraadi taotletud.
Töös kasutatud kõik teiste autorite materjalid on varustatud vastavate viidetega.

Töö valmis..... juhendamisel
“.....”.....201….a.
Töö autor
..... allkiri

Töö vastab bakalaureusetööle esitatavatele nõuetele.
“.....”.....201….a.
Juhendaja
..... allkiri

Lubatud kaitsmisele.
..... õppekava kaitsmiskomisjoni esimees
“.....”.....201… a.

..... allkiri

TTÜ mehhatroonikainstituut

Mehhatroonikasüsteemide õppetool

Bakalaureusetöö ülesanne

2014 aasta kevadsemester

Üliõpilane: Madis Taimre, 105663MAHB
Õppekava MAHB02/09
Eriala mehhatroonika
Juhendaja: Maido Hiiemaa, teadur
Konsultandid:

BAKALAUREUSETÖÖ TEEMA:
(eesti keeles) Tarkvara 3D printerit juhtivale mikrokontrollerile
(inglise keeles) Software for the Central Microcontroller of the 3D printer

Lõputöös lahendatavad ülesanded ja nende täitmise ajakava:

Nr	**Ülesande kirjeldus**	**Täitmise tähtaeg**
1.	Teha ülevaatlik kokkuvõte levinumate 3D printerite tööpõhimõtetest, struktuurist ja juhtimisest.	15.02.2014
2.	Koostada konkreetse 3D printeri funktsionaalsusele vastav juhtprogrammi kirjeldus.	01.03.2014
3.	Valida mikrokontroller, arendustarkvara ja arendustöö aluseks võetavad lähtetekstid/draiverid	15.03.2014
4.	Kirjutada näidisprogramm, mis väljastab samm-mootorite juhtsignaale nt. SD kaardile laetud failide alusel.	01.05.2014
5.	Lõputöö köidetud ja valmis esitamisele.	21.05.2014

Lahendatavad insenertehnilised ja majanduslikud probleemid: Töö eesmärgiks on luua osa 3D printeri juhtimise tarkvarast ja anda selge ülevaade moodulitest, mis tuleks täisfunktsionaalse 3D printeri väljatöötamiseks juurde teha. Väljapakutav riistvara peab võimaldama telgi juhtida sünkroonselt.

Täiendavad märkused ja nõuded: Peab näitama ostsilloskoobilt, et programm saadab käske sünkroonselt.

Töö keel: eesti

Kaitsmistaotlus esitada hiljemalt 12.05 **Töö esitamise tähtaeg** 22.05

Üliõpilane Madis Taimre /allkiri/ kuupäev

Juhendaja…… /allkiri/ ……. kuupäev……

SISUKORD

EESSÕNA

Teema sai valitud juhendaja Maido Hiiemaa soovitusel. Töö on edasiarendus aastal 2012 tehtud 3D printeri projektile, mida tegid Martin Parker ja Holger Kruusa. Samuti on aluseks võetud aastal 2013 tehtud Martin Rannamäe bakalaureusetöö selle projekti raames, millest on saadud suurem osa algandmeid.

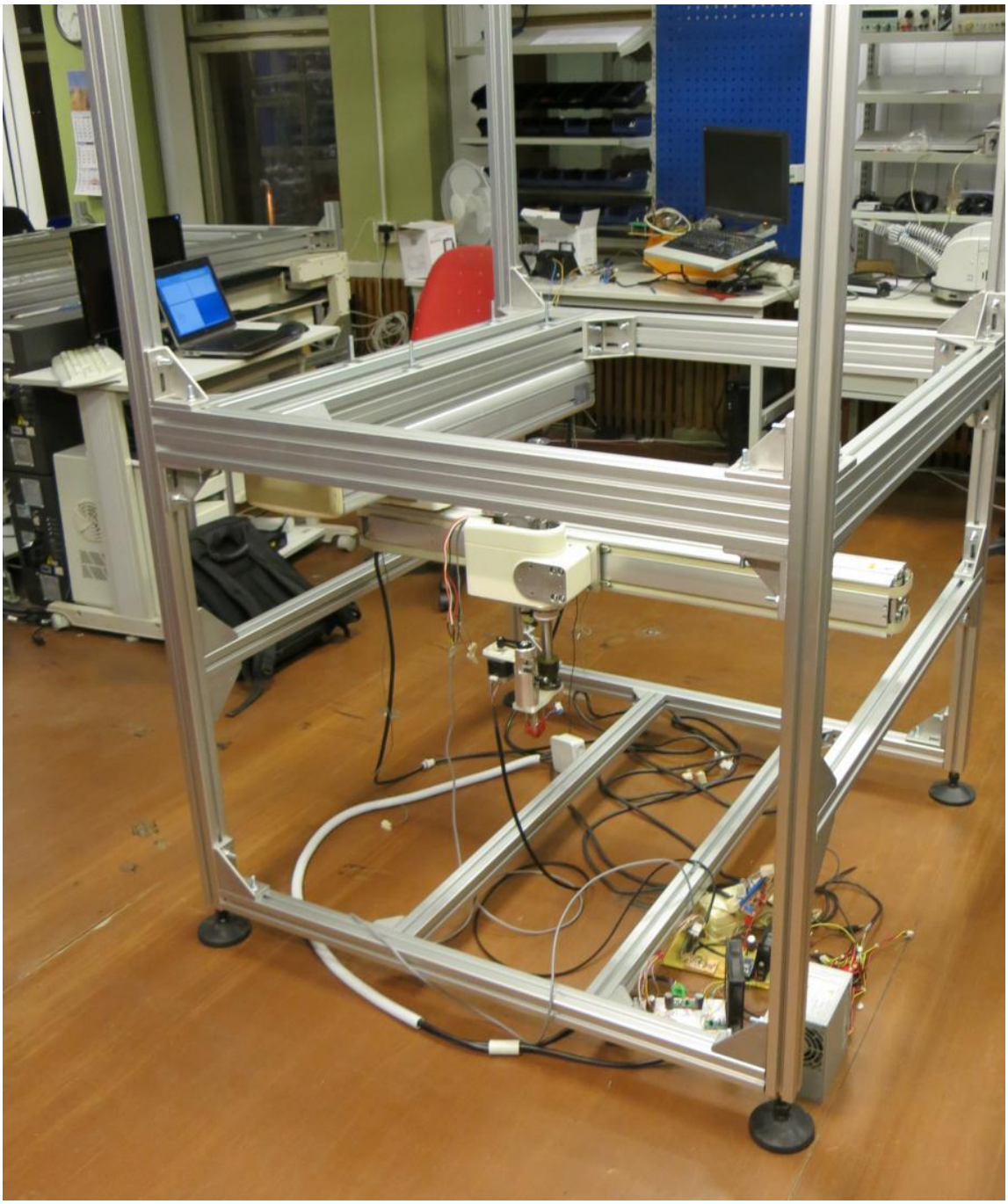
1. SISSEJUHATUS

Töö sai valitud, sest sellel on suur rõhk programmeerimisel, millest on autor huvitatud. Samuti andis see võimaluse autoril osa programmist valmis kirjutada ja seeläbi koodi kirjutamist õppida. Autor on varem samm-mootorite käima panemiseks koodi kirjutanud ja nende juhtimisega mingil määral tuttav.

3D printimine on viimaste aastatega väga oluliseks teemaks muutunud. Seda seetõttu, et peaaegu kõike saab sisse skaneerida, muuta ja seejärel välja printida. Seepärast on 3D printimisel potentsiaali tohutult areneda. 2013 aasta alguses ütles USA president Barack Obama 3D printimise kohta, et „sellel on potentsiaali revolutsioneerida viisi, kuidas me teeme ükskõik mida“[6] ja teatas plaanist tekitada võrgustik viieteistkümnest 3D printimist arendavast firmast USA-s, mis peaksid seda tööstust edasi viima. Samuti said 2013 aasta alguses läbi SLS (*selective laser sintering*) põhilised patendid, mis tekitas võimaluse 3D printimist põhjalikumalt arendama ja uuendama hakata. Seetõttu muutuvad printerid arvatavasti efektiivsemaks ja odavamaks. Samuti võivad need muuta hulgitootmist. Näiteks Lego arendab võimalust, kuidas oma mänguasju massiliselt toota kasutades 3D printimist. Sellest kõigest võib järeldada, et lähitulevikus on oodata 3D printimise tähtsuse suurenemist.[6]

Ülesande uurimuslikuks osaks on leida lahendus 3D printeri juhtmooduli programmile. Uurida võimalikke probleeme, mis tekivad suurema printeri juhtimisel ja pakkuda programmiline lahendus neile. Samuti välja selgitada juhtprogrammi sisendid ja väljundid ning jaotada programm moodulitesse. Teha moodulid nii, et neid saaks programmiliselt kõigepealt lihtsamalt kirjutada ja hiljem, kui on 3D printer tööle saadud, põhjalikumalt. 3D printeri juhtimisel on väga suur rõhk samm-mootoritel ja seetõttu tuleb nende tööpõhimõtet ja juhtimist põhjalikult uurida. Uurimuslik osa tuleb teha nii, et seda saavad hiljem üliõpilased kasutada 3D printimise ja samm-mootorite juhtimise õppimiseks ja Mehhatroonikainstituudi laboris oleva printeri edasi arendamiseks. Töö praktilise osa eesmärgiks on kirjutada osa koodist 3D printerit juhtivale mikrokontrollerile nii, nagu on kirjeldatud töö uurimuslikus osas.

Mehhatroonikainstituudi laboris olev printeril on olemas raam, printeri pea, Hirata karteesiuse roboti teljed ja ajutine elektroonika. Selle töötamist on osaliselt katsetatud kasutades tarkvara *Smoothie*. Printeril katsetati telgede liikumist ja printeri pea töötamist. Eseme printimiseni ei jõutud.



Sele 1.1. Mehhatroonikainstituudis olev 3D printer [13, lk 34]

Saadaval olevaid 3D printeri juhtmoodulite tarkvarasid on palju. Nendeks on näiteks *Sprinter*, *Teacup*, *sjfw*, *Marlin*, *Sailfish*, *Makerbot*, *Grbl*, *Repetier-Firmware*, *aprinter*, *RepRap Firmware* ja *Smoothie*. Antud töös on uuritud *Smoothie* põhimõttelist ülesehitust ja sellest on saadud ideid, kuidas võiks programm mooduliteks jaotada.

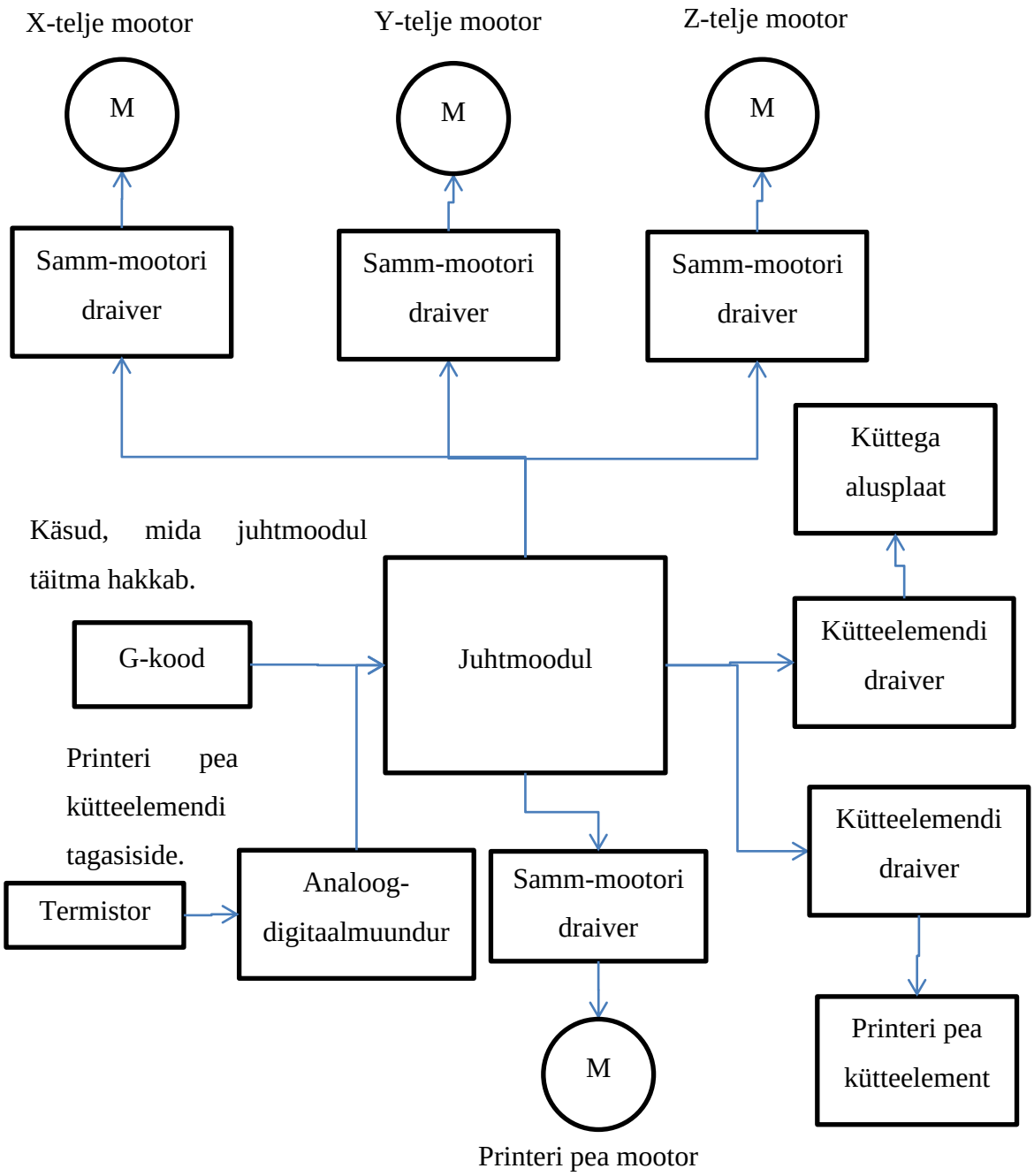
Töö jaoks info kogumisel ja 3D printeri tööpõhimõtte välja selgitamisel on kasutatud F-M (*Function-Means*) puud. Töö tegemisel on kasutatud aja planeerimist koos tähtaegadega. Koodi kujutamiseks on kasutatud voodiagrammi. Töö praktilises osas on kasutatud Olimexi mikrokontrollerit LPC-P2148, Seggeri programmaatorit J-Link seerianumbriga 268003858 ja

programmi SEGGER J-Link GDB Server V4.80c. Veel lisaks on kasutatud programmi Eclipse Standard/SDK Kepler Service Release 1 versiooni ja GNU ning YAGARTO töövahendeid.

Töö teises peatükis on analüüsitud Mehhatroonikainstituudis olevat printerit ja mida programm peab tegema, et seda juhtida. Peatüki lõpus on saadud lähtemäärang. Kolmandas peatükis on kõigepealt põhjalikumalt uuritud samm-mootorite juhtimist ja analüüsitud erinevaid juhtimise lahendusi ning kuidas seda osa programmis moduleerida. Seejärel on kirjas sellest, kuidas printeri juhtimiseks mõeldud käsud juhtmoodulisse saada, telgede piirlülititest, küttega alusplaadi ja printeri pea juhtimisest. Neljandas peatükis on toodud välja kokkuvõttev lõplik lahendus ja viiendas on kirjas, kui palju on programmist valmis kirjutatud.

2. LAHENDATAVA PROBLEEMI OLEMUS

2.1 3D printer



Sele 2.1. 3D printeri lihtsustatud ülesehitus

3D printimine on protsess, kus objekt luuakse kihthaaval nii, et materjal sulatatakse kokku üheks kehaks [8].

Antud 3D printer kasutab sulatatud sadestumise vormimise viisi (Fused Deposition Modeling). Seda tüüpi printer kasutab materjalina termopolümeeri, mis on keritud poolile. Materjali kuumutatakse sulatamistemperatuurile ja surutakse läbi otsiku prinditava objekti peale.[14, lk 10]

2.1.1 Raamistik

Printeri raamistikuks on alumiiniumist raam, millele on kinnitatud nelja teljeline Hirata karteesiuse robot. Kolm telge lubavad liigutada materjali väljalaske pead igas suunas ja neljas pöörata seda ümber üles-alla liikuva telje, aga seda ei rakendatud süsteemi. Programm peab seega juhtima kolme mootorit.[14, lk 15]

Raam on mõõtmetega 1150x1150x1000 mm ehk palju suurem, kui tavalisel poest ostetaval printeril [14, lk 15]. Seega peab arvestama telje liigutamisest tekkiva inertsiga ja arvestama, et see võib tekitada raamistikus ülekoormusi ning vibratsioone. Kui inerts kahjustab süsteemi, siis programm peab kiirendama ja aeglustama mootoreid sujuvalt.

2.1.2 Mootorid

Kahel teljel on bipolaarsed samm-mootorid:

- täissamm 1,8 kraadi;
- maksimaalne moment 2,2 Nm;
- 2-faasi;
- 3A faasi kohta;
- 12V pingelang mähistel [14, lk 17].

Ühel teljel on NEMA 23 tüüpi bipolaarne samm-mootor:

- täissamm on 1.8 kraadi;
- maksimaalne moment on 1,27 Nm;
- 2-faasi;
- 3A faasi kohta [14, lk 17].

Selle informatsiooni põhjal peab uurima erinevaid lahendusi samm-mootorite juhtimiseks, analüüsima neid ja kindlaks määrama oma programmi väljundi. Mootoritel tagasiside puudub ja seega on mootorite juhtimine avatud süsteem.

Programm peab arvutama, kui palju liigutada mootoreid, kasutades täissammu nurka.

2.1.3 Piirlülitid

Piirlülitid määravad ära telgede piirasendid. Programm peab olema võimeline ära hoidma olukorda, kus teljed hakkavad liikuma oma piiridest välja ja seega võivad kahjustada süsteemi. Samuti peab juhtplaat enne printimise algust liigutama printeri pea algasendisse, sest selle kaudu teeb programm arvutused, kui palju peab telgi liigutama. Seega lülititelt saadud informatsioon on üheks programmi sisendiks.

2.1.4 Küttega alusplaat

Printerile tuleb lisada küttega alusplaat, mis hoiab ära prinditava detaili liiga kiire kahanemise, mis võib tekitada sisepingeid [14, lk 19].

Juhtmoodul peab olema võimeline alusplaadi temperatuuri hoidma ja seda sisse ning välja lülitama. Alusplaadi peale tuleb paigaldada termistor, mis annaks tagasisidet temperatuuri kohta.[14, lk 20]

2.1.5 Printeri pea

Projektis on kasutusel *3D Touch* printeri pea. Selle ülesandeks on juhtida vedelat materjali alusplaadile, et tekiks sellest materjalist prinditav objekt. Materjali juhitakse rullikute abil, mida paneb liikuma samm-mootor. Materjali vedelaks tegemiseks on printeril kuumutuspea, mille ülesandeks on soendada rullikutelt tulevat materjali. Samuti on kuumutuspeal termistor, millega see annab tagasisidet juhtmoodulile. Samm-mootorite ja kütteelemendi juhtimiseks on olemas elektroonika plaat. Seega peab programm andma signaale printeri pead juhtivale elektroonika plaadile.[13, lk 17]

2.2 Süsteemi juhtimine

Süsteemi juhtimiseks on olemas RS274/NGC standard, mille on välja töötanud NIST (*National Institute of Standards and Technology*) [10]. See standard määrab ära käsud ja nende täitmise viisid NC (*Numerical control*) masinate juhtimiseks. Programm peab suutma käske täita sellest standardist lähtuvalt. Tervet RS274/NGC-d ei ole mõtet programmi rakendada, sest süsteemil puuduvad seadmed kõikide käskude jaoks. Lisaks eelnimetatud standardile võib rakendada vastavalt vajadusele muid käske, mis on 3D printeritele omased.

2.2.1 RS274/NGC keel (G kood)

RS274/NGC keeles (kutsutakse ka G-koodiks) on käsud jaotatud ridadeks, mis kokku pannes moodustab programmi. Tüüpiline rida võib alata reanumbriga (ei ole kohustuslik), millele järgneb üks või mitu „sõna“, mis annab käsu või argumendi käsule. Sõna koosneb tähest, millele järgneb number. Näiteks rida „G1 X90“ tähendab, et liiguta midagi sirges joones X-i üheksakümnenda koordinaadini.[10]

G-koodi võib ise kirjutada või lasta genereerida programmil, mida kutsutakse *Sliceriks*.

Juhtmoodulil peab seega olema liides, mille kaudu G-kood moodulisse saada, ja see tähendab, et programmis peab olema vastava liidese draiver. Samuti peab programm suutma G-koodi parsida ja kontrollida, kas see vastab standardile.

2.3 Lähtemäärang

Lahendatava probleemi olemusest saab programmi lähtemäärangu:

- juhib sünkroonselt kolme bipolaarset samm-mootorit;
- kiirendab ja aeglustab mootoreid nii, et see ei kahjustaks süsteemi;
- peab viima printeri pea algasendisse;
- ära tuvastama olukorra, kui ekstruuder hakkab telgedest välja liikuma ja selle ära hoida;
- juhtima küttega alusplaati;
- loeb juhtmoodulisse ja parsib G koodi;

- täidab käske vastavalt NIST RS274/NGC standardile;
- juhtima ekstruuderit.

3. 3D printeri juhtimine

3.1 Samm-mootorite juhtimine

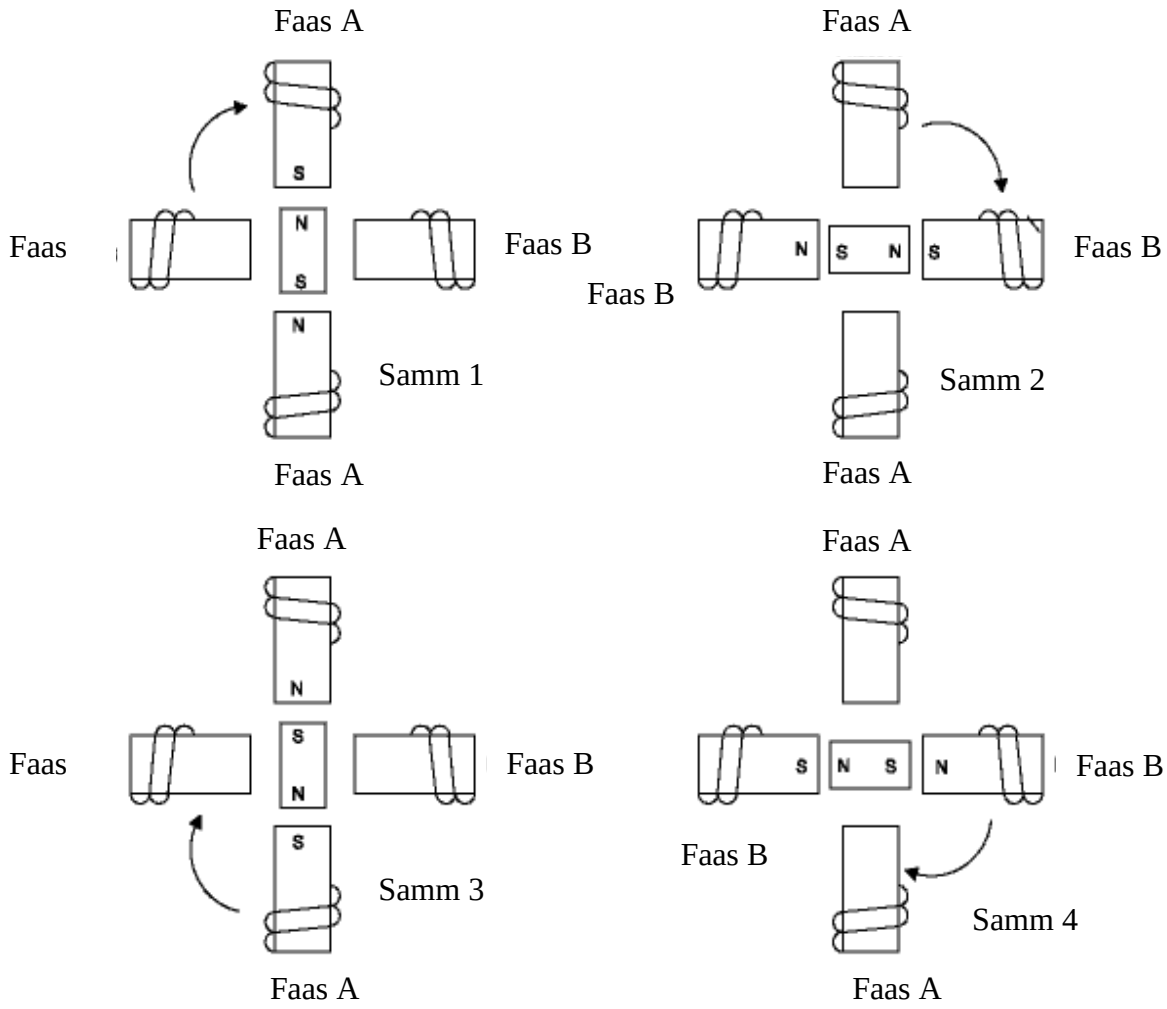
Samm-mootor muudab alalispinge impulsid mootori võlli pöördliikumiseks. Ehitusviisilt on nad sünkroonmootorid, mille rootor pöörleb vastavalt staatorimähisesse antud taktimpulssidele. Seda kujutab sele 3.1.

Igale mähistele antud impulsile vastab teatud pöördenurk ja seega saab mootoreid juhtida ka ilma tagasisideta. Kuna antud seadmel mootoritel tagasiside puudub, siis antud töös on uuritud ainult avatud ahelaga süsteeme.

Samm-mootorid jagunevad ühenduse viisist sõltuvalt uni- ja bipolaarseteks. Projektis kasutatud mootorid on bipolaarsed.

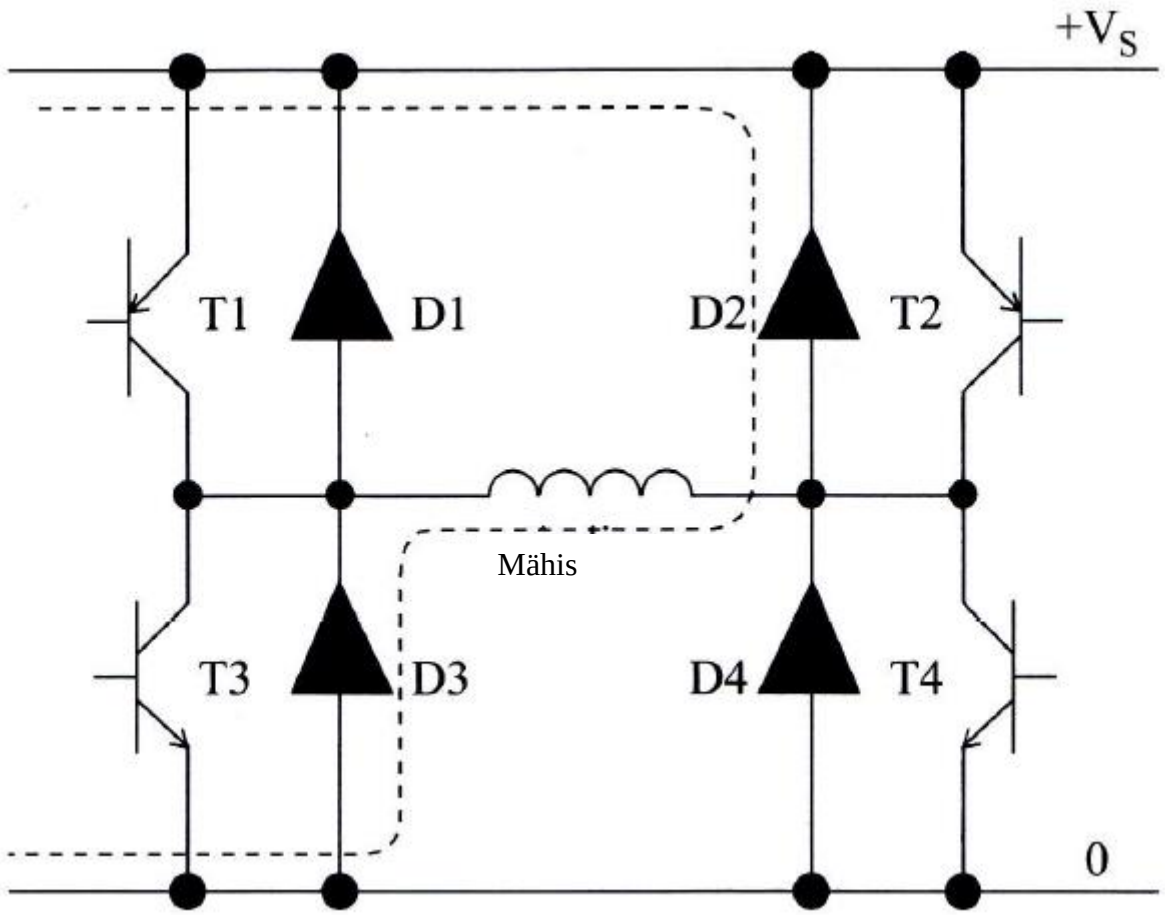
3.1.1 Bipolaarsete mootorite juhtimise põhimõte

Püsimagnetiga mootoritel on kaks mähist, magnetiseeruv staator ja rootoriks püsimagnet. Mähisele voolu andes tekib staatoris magnetväli põhja- ja lõunapoolusega, mis liigutab rootorit. Vaheldumisi voolu andes mähistele hakkab rootor pöörlema.[18]



Sele 3.1. Samm-mootori liigutamise lihtsustatud skeem [18]

Bipolaarse mootori juhtimiseks peab olema võimalik mähisest voolu mõlemat pidi läbi lasta. Selleks kasutatakse H-sildu. Seles 3.2 on näidatud ühe faasi juhtimiseks mõeldud skeem.



Sele 3.2. Bipolaarse samm-mootori ühe faasi juhtimiseks mõeldud skeem [2, lk 17]

Transistore lülitatakse paarides vastavalt sellele, mis pidi peab olema voolu polaarsus. Mähise positiivseks pingestamiseks lülitatakse transistorid T1 ja T4 sisse, et vool liiguks vooluallikast läbi transistori T1 mähisesse ja seejärel läbi transistori T4 tagasi vooluallikasse. Selleks, et tekitada mähises teistpidine polaarsus, lülitatakse sisse transistorid T2 ja T3.[2, lk 16]

Transistore on vaja selleks, et võimendada juhtsignaale ja dioodid eemaldavad mähises indutseeritud voolu, kui transistorid välja lülitada. Seles 3.2 on näidatud voolu liikumine, kui transistorid T1 ja T4 on välja lülitada.[2, lk 17]

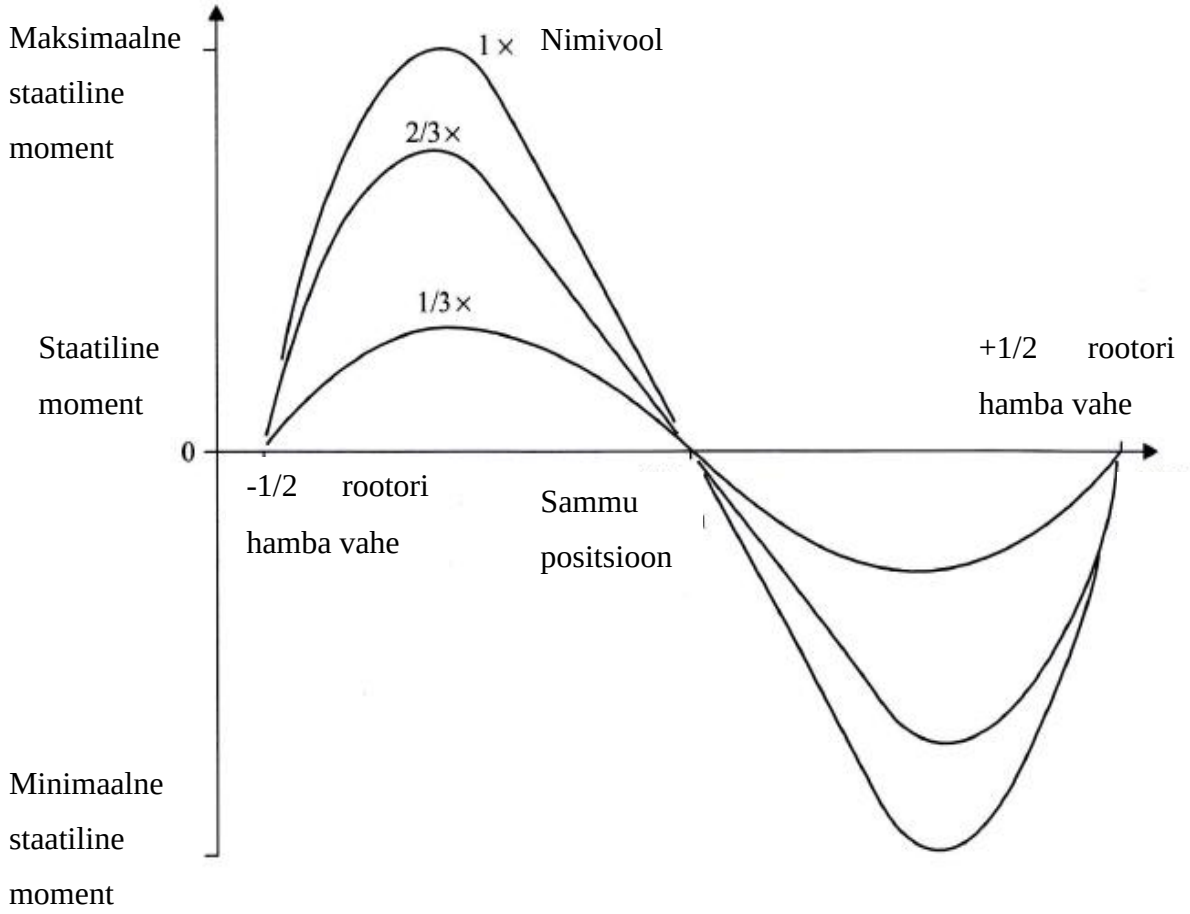
3.1.2 Mootori poolt tekitatav moment ja printeri pea positsioneerimine

Selleks, et printeri pead täpsemalt positsioneerida, peab uurima staatilisi momente samm-mootoris. Koormused tekitavad väikest positsioneerimise ebatäpsust staatilises olekus.

Mootor peab tekitama piisavalt pöördemomenti, et tasakaalustada koormusest tekkinud momenti ja seega on rootor teise kraadi all eeldatavast sammu positsioonist. See staatiline positsiooni viga sõltub süsteemi välisest pöördemomendist, aga on sõltumatu eelnevatest sammudest ja seega mitte kumulatiivne.[2, lk 25]

Paljudel juhtudes staatilist viga saab vähendada sellega, et mitu mootori faasi pingestatakse samal ajal. Printeri pea täpsemaks positsioneerimiseks võiks arendatav programm seda võimalust lubada.

Staatilist viga saab uurida mootori staatilise pöördemomendi/rootori positsiooni graafikust. See näitab mootori poolt arendatavat momenti funktsioonina rootori positsioonist.[2, lk 25]



Sele 3.3. Pöördemomendi/rootori positsiooni graafik [2, lk 26]

Sele 3.3 näitab, et kui staatori ja rootori hambad on ühel joonel, siis arendatav jõumoment on null. Positiivne jõumoment tekib siis, kui rootor on nihkunud negatiivses suunas ja negatiivne jõumoment tekib siis, kui rootor on nihkunud positiivses suunas.[2, lk 26 - 27]

Kui koormata mootorit mingi jõumomendiga, siis rootor liigub positsioonile, kus koormuse moment on sama, mis mootori poolt arendatav moment. Sellest tekibki staatiline viga ja selest 3.3 on näha, et see ei saa olla suurem, kui pool rootori hammaste vahet (eeldades, et mootor ei ole jätnud samme vahele).[2, lk 27]

Staatilist viga saab hinnata, kui pöördemomendi/rootori positsiooni funktsiooni lähendada siinusele [2, lk 27]. Mootori arendatav pöördemoment on seega ligikaudu:

$T = -T_{PK} \cdot \sin(p \cdot \theta)$ [2, lk 17], (3.1)

kus T – arendatav pöördemoment,

T_{PK} – mootori maksimaalne pöördemoment,

p – rootori hammaste arv,

θ – sammu positsioon.

Kui mootorile lisada koormus, siis rootor nihkub. Kuna mootori poolt arendatav pöördemoment on sama koormuse poolt tekitatud pöördemomendiga [2, lk 28], siis saab kirjutada:

$T_L = T = -T_{PK} \cdot \sin(p \cdot \theta_e)$ [2, lk 28], (3.2)

kus T_L – koormuse poolt tekitatud pöördemoment,

θ_e – rootori nihe (staatiline posistiooni viga).

Siit valemist staatiline positsiooni viga on

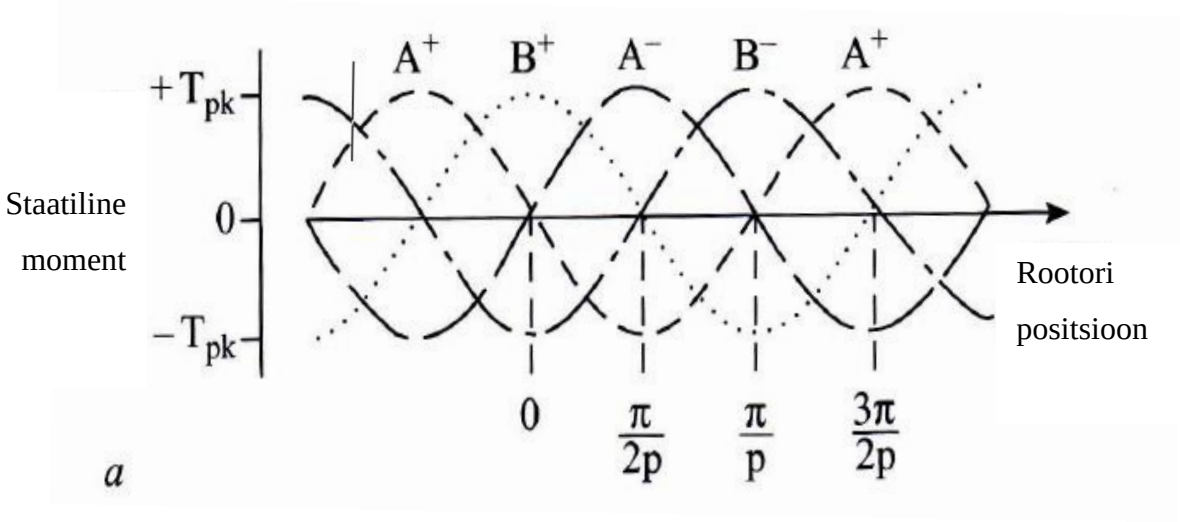
$$\theta_e = \frac{\sin^{-1}\left(-\frac{T_L}{T_{PK}}\right)}{p}$$ [2, lk 28] (3.3)

Sellest valemist on näha, et viga saab vähendada, kui suurendada mootori poolt arendatavat maksimaalset pöördemomenti, vähendada koormust või suurendada hammaste arvu. Sele 3.3 graafikust on näha, et kui juhtida mootoreid faasi nimivoolust madalama vooluga, siis maksimaalne pöördemoment väheneb ja järelikult staatiline viga suureneb. Seega peaks juhtima mootoreid faasi nimivooluga.

Staatilise vea vähendamiseks on samm-mootoritel erinevad **juhtimisviiside talitlused**. Kõige lihtsam viis on pingestada ainult ühte mähist korraga, mida nimetatakse poolsamm-talitluseks.

Mitut mähist korraga pingestamist nimetatakse täissamm-talitluseks. Sellisel juhul suurendatakse mootori poolt arendatavat maksimaalset pöördemomenti. Samuti on võimalik piirata voolutugevust mähistes ja sellega tekitada palju vahepealseid samme, mida nimetatakse mikrosamm-talitluseks. See oleks võrdväärne valemi 3.3 järgi hammaste arvu suurendamisega.

Poolsamm-talitluse joonist näitab sele 3.4. Sellel on kujutatud projektis ka kasutusel oleva bipolaarse mootori pingestamist. Bipolaarsel mootoril on kaks faasi, mida saab pingestada positiivselt või negatiivselt. Kui igat faasi kordamööda pingestada, siis on vaja nelja sammu, et mootor liiguks ühe hammaste vahe kauguse. Seega üks samm on võrdne neljandiku hamba vahe pikkusest ja iga faasi jaoks pöördemomendi/rootori positsiooni funktsiooni graafikud on nihkes selle kauguse võrra nagu kujutab sele (3.4).[2, lk 33]



Sele 3.4. Poolsamm-talitluse faaside pingestamine [2, lk 34]

Vastavalt selele 3.4 saab mootori poolt arendatava pöördemomendi välja kirjutada iga faasi kohta nii:

$T_{A+} = -T_{PK} \cdot \sin(p \cdot \theta)$ [2, lk 33], (3.4)

kus T_{A+} – pöördemoment, kui mootori A mähist pingestada positiivselt.

$T_{A-} = -T_{PK} \cdot \sin(p \cdot \theta - \pi)$ [2, lk 33], (3.5)

kus T_{A-} – pöördemoment, kui mootori A mähist pingestada negatiivselt.

$$T_{B+} = -T_{PK} \cdot \sin\left(p \cdot \theta - \frac{\pi}{2}\right) \quad \text{[2, lk 33]}, \tag{3.6}$$

kus T_{B+} – pöördemoment, kui mootori B mähist pingestada positiivselt.

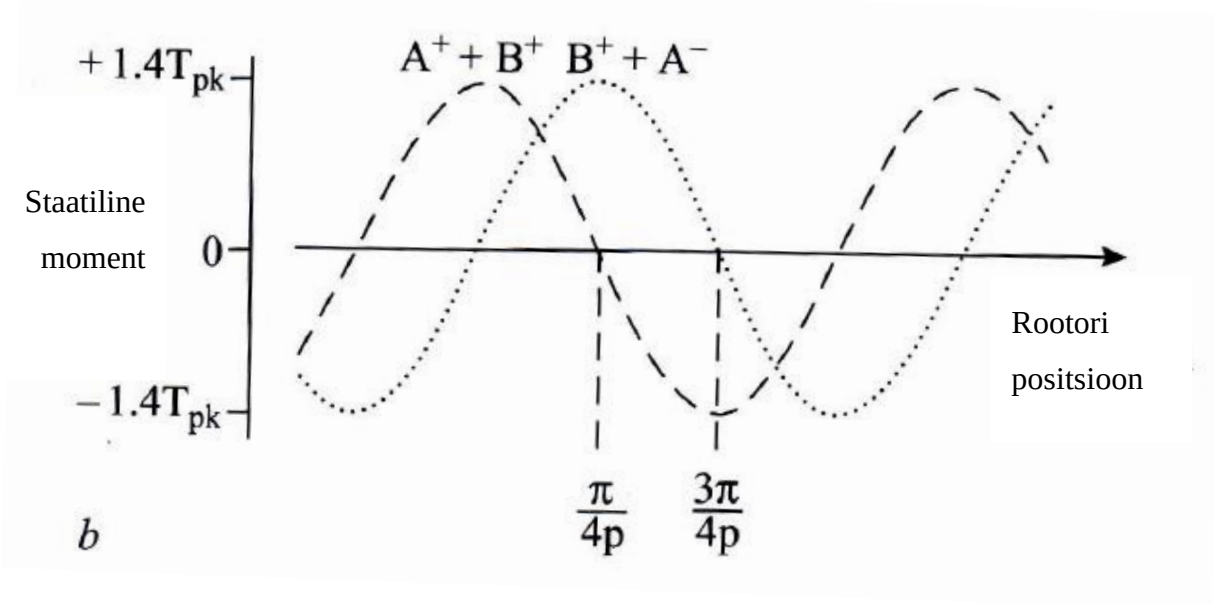
$$T_{B-} = -T_{PK} \cdot \sin(p \cdot \theta - \frac{3\pi}{2}) \quad \text{[2, lk 33]}, \tag{3.7}$$

kus T_{B-} – pöördemoment, kui mootori B mähist pingestada negatiivselt.

Täissamm-talitlus on võimalik seepärast, et samm-mootoril on faasi mähised elektriliselt isoleeritud ja neid juhivad erinevad vooluahelad ning seetõttu on neid võimalik samaaegselt pingestada [2, lk 29]. Seles 3.5 on näidatud seda, kui kahte faasi samal ajal pingestada. Eelnevate valmite põhjal saab välja arvutada, kui palju suurem on tekkinud maksimaalne pöördemoment.

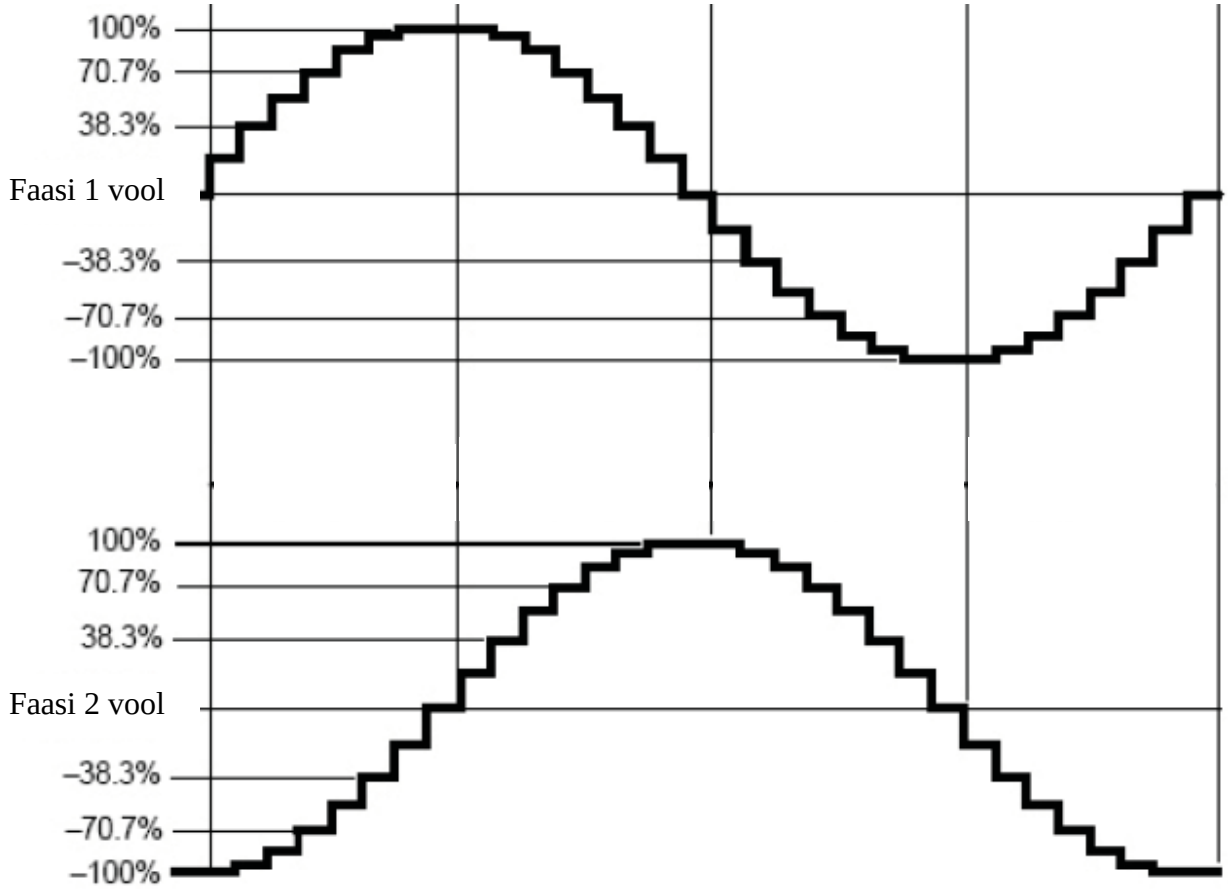
$$\begin{aligned} T_{A+} + T_{B+} &= -2T_{PK} \cdot \sin\left(p \cdot \theta - \frac{\pi}{4}\right) \cdot \cos\left(\frac{\pi}{4}\right) \\ &= -1.4 \cdot T_{PK} \cdot \sin\left(p \cdot \theta - \frac{\pi}{4}\right) \end{aligned} \quad \text{[2, lk 34]} \tag{3.8}$$

Teiste faaside kombinatsioonidega on valem sarnane. Seega kahte mähist korraga pingestades tõuseb jõumoment 1.4 korda. Suurem jõumoment tekitab väiksema staatilise vea. Samal ajal tuleb arvestada sellega, et nii kulub rohkem energiat ja toiteallikas peab olema kaks korda võimsam.[2, lk 34]



Sele 3.5. Täissamm-talitluse puhul faaside pingestamine [2, lk 34]

Veel lisaks on olemas talitlusviis, mis jagab terve sammu paljudeks väikesteks sammudeks, mida kutsutakse **mikrosammudeks**. See saavutatakse nii, mähistest lastakse erineva suurusega voole läbi. Need n-ö voolutasemed on kõik mingi suuruse võrra väiksemad nimivoolust. Sele 3.6 on kujutatud bipolaarse mootori faasi voole, kui on kasutatud üks kaheksandik mikrosamm-talitlust.[2, lk 34 - 35]



Sele 3.6. 1/8 mikrosamm-talitluse faasi voolud [12]

Selgituseks võib vaadata samm-mootorit, mille maksimum arendatav pöördemoment on võrdeline faasivooluga. Kui mootori maksimaalne pöördemoment on võrdeline vooluga ($T_{PK} \cong k_T \cdot i$, kus k_T on võrdetegur ja i on vool läbi mähise), siis saab kirjutada:

$T_A = -k_T \cdot i_A \cdot \sin(p \cdot \theta)$ [2, lk 35], (3.9)

kus i_A – vool läbi mägise A.[2, lk 34 - 35]

$T_B = -k_T \cdot i_B \cdot \sin\left(p \cdot \theta - \frac{\pi}{2}\right)$ [2, lk 35], (3.10)

kus i_B – vool läbi mägise B.

Mootorite juhtplaat võib omakorda tekitada voolu, mis on osa mootori nimivoolust:

$i_A = I \cdot \cos(\alpha)$ [2, lk 35], (3.11)

kus i_A – vool läbi mähise A,

I – mootori nimivool,

α – muutuja, millega vähendada faasi voolu. [2, lk 35]

$i_B = I \cdot \cos(\alpha - \frac{\pi}{2})$ [2, lk 35], (3.12)

kus i_B – vool läbi mähise B.

Mootori tekitatav pöördemoment oleks seega:

$$T = T_A + T_B = -k_T \cdot I \cdot [\sin(p \cdot \theta) \cdot \cos(\alpha) + \sin\left(p \cdot \theta - \frac{\pi}{2}\right) \cdot \cos\left(\alpha - \frac{\pi}{2}\right)] = -k_T \cdot I \cdot \sin(p \cdot \theta - \alpha)$$ [2, lk 35] (3.13)

Tasakaalu koht ilma koormuseta oleks seega $\theta = p \cdot \theta - \alpha$. Järelikult rootorit saaks paigutada ükskõik millisesse positsiooni, kui mähistele antakse voolu erinevates suurustes [2, lk 35]. Sellisel juhtplaadil oleksid erinevad voolu tasemed. Mikrosammu pikkuse saab arvutada valemiga:

$\theta_M = l/(2 \cdot N_L)$ [2, lk 35], (3.14)

kus θ_M – mikrosammu pikkus,

N_L – erinevate voolutasemete arv,

l – rootori hammaste vahe.

Mikrosamm-talitluse boonuseks on see, et sellega vähendatakse mootori jõnksatuslikku liikumist madalatel kiirustel ja probleeme resonantsiga [9].

Mikrosamm-talitlusel on mitmeid probleeme. Tavapärase juhtimise talitluse korral on mootori tasakaalupunktid sõltuvad staatori ja rootori paiknemisest ja mitte voolu tasemest. Mikrosamm-talitluse puhul on tasakaalupunktid oluliselt sõltuvad voolust ja iga viga voolutugevuses kandub positsioneerimise ebatäpsusesse.[2, lk 35]

Kõige suurem probleem mikrosamm-talitlusega on see, et kui suurendada mikrosammude arvu, siis mootoril ei pruugi olla piisavalt jõudu, et igat sammu sooritada. Selest 3.3 on näha, et pöördemoment sõltub rootori positsioonist. Seega kui sammude vahel on palju mikrosamme, siis rootor ei pruugi olla liikunud piisavalt kaugele, et tekkinud moment ületaks kõik vastukäivad momendid (koormusest ja hõõrdetakistusest tekkinud momendid). Selle tulemusel jätab mootor järgmise sammu tegemata ja ootab, millal mähiste pingestamise järjekord on jõudnud kaugemale. Järelikult sammude eraldusvõime tõuseb, aga täpsus võib langeda. Eriti muutub see probleemiks siis, kui peaks mootorit teistpidi pöörlema panema. Sellisel juhul võib minna väga mitu sammu, kuni mootor viimaks liikuma hakkab. Tabelis 3.1 on näidatud mikrosammude arv täissammude vahel ja hoidemoment mikrosammu kohta.[11]

Tabel 3.1. Hoidemoment erinevate mikrosammudeks jagamise korral [11]

Mikrosamm/täissamm	Hoidemoment/mikrosamm
1	100,00 %
2	70,71 %
4	38,27 %
8	19,51 %
16	9,80 %
32	4,91 %
64	2,45 %
128	1,23 %
256	0,61 %

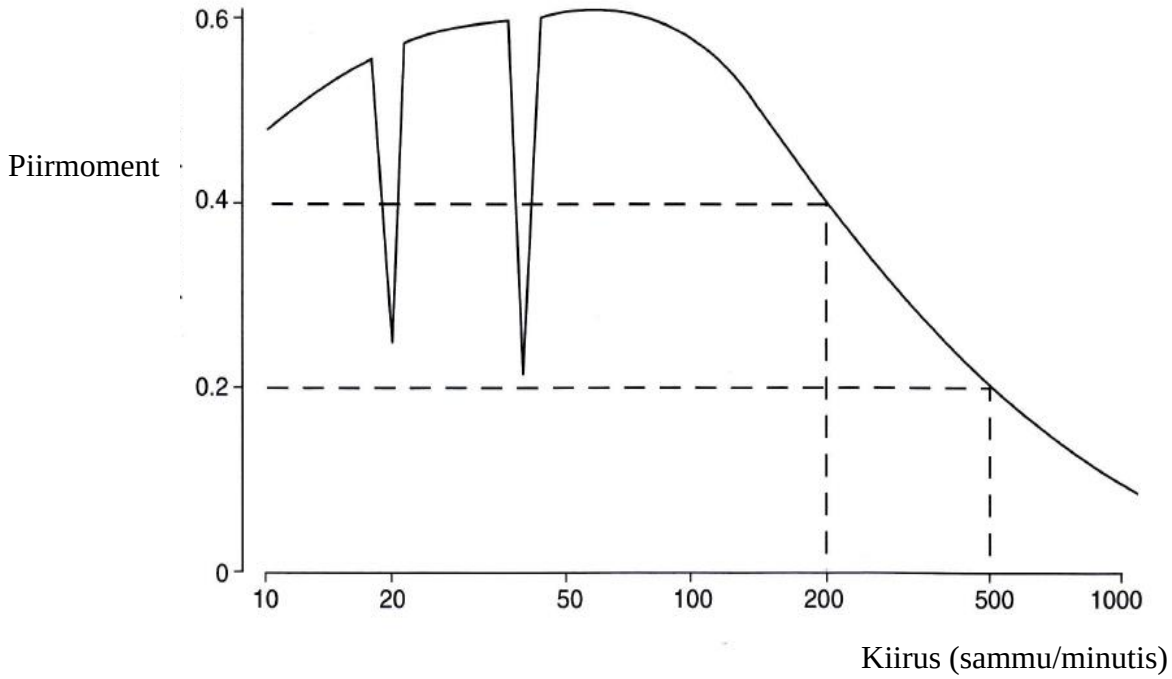
Antud projekti jaoks **samm-talitluse valiku** saab teha alles siis, kui on välja selgitatud kõik mootorile vastu käivad pöördemomendid. Seda saab teha kõige paremini katsetades. Kõige lihtsam on poolsamm-talitlus. Kui tuleb välja, et koormus on väga suur ja rohkem on vaja tekitada pöördemomenti, siis tasuks kasutada täissamm-talitlust. Kui vastu käib koormus ei ole kuigi suur ja mootor hakkab tegema samme tehes jõnksatusi ning soovitakse suurendada positsioneerimise täpsust, siis võib kasutada mikrosamm-talitlust. Selle talitluse kasutades peab seejuures kindlaks tegema, et mootoril on piisavalt jõudu, et iga samm sooritada.

Majandusliku poole pealt on kõige kallim rakendada mikrosamm-talitlust, sest see nõuab palju kallimat samm-ajamit, mis suudaks tekitada erinevad voolu tasemeid [2, lk 36].

Täissamm-talitluse puhul läheb kaks korda rohkem energiat, kui poolsamm-talitlust kasutades. Samuti on vooluallikas võimsam ja kallim. Kõige odavam on seega rakendada poolsamm-talitlust.

3.1.3 Maksimaalne kiirus

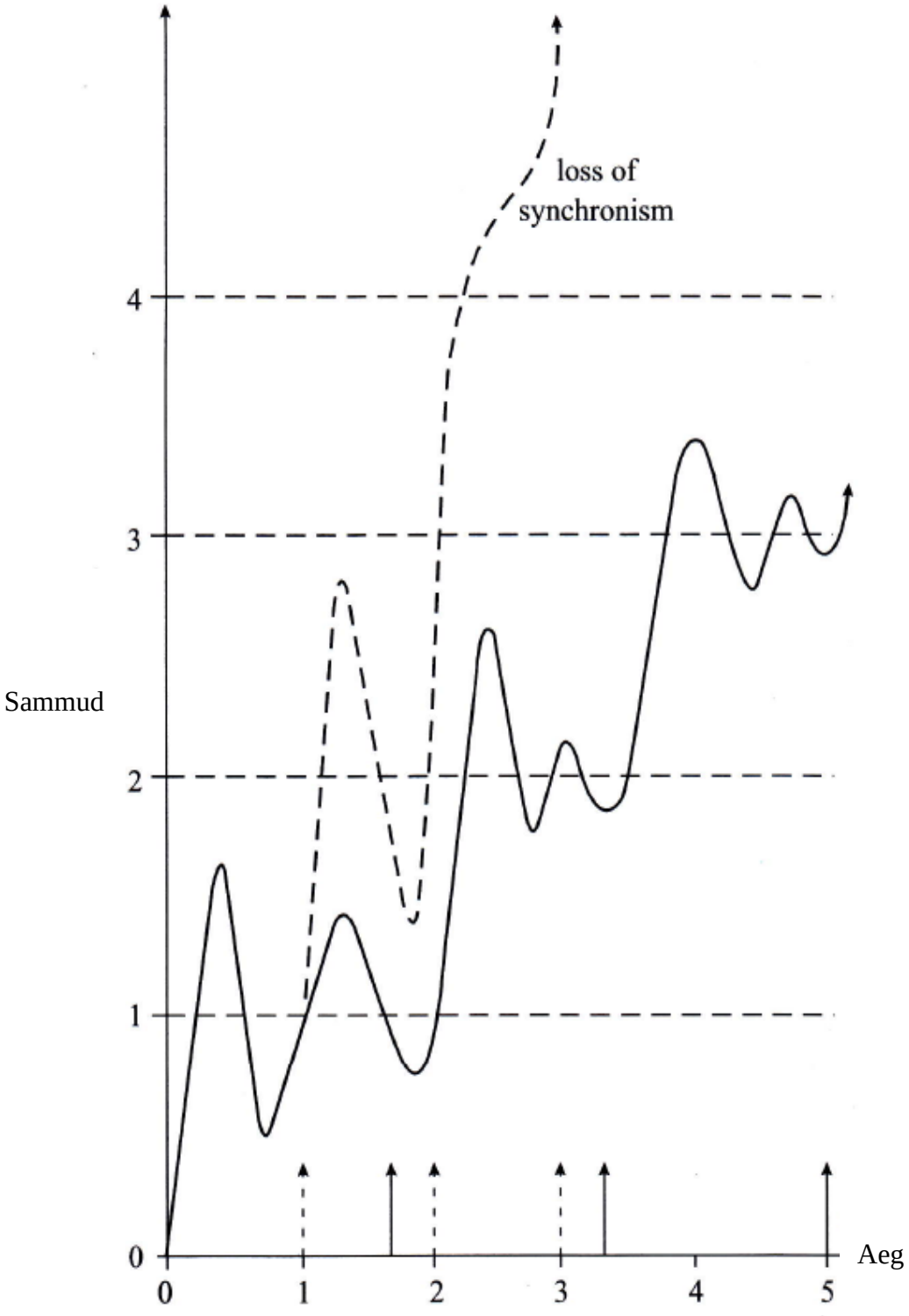
Programm peab teadma, mis kiiruseni on võimalik mootorit ajada, et see suudaks koormust edasi viia. Selle informatsiooni saab mootori piirmomendi/kiiruse graafikult. Piirmoment on väärtus, milleni saab mootorit koormata ilma samme kaotamata [15]. Tüüpiline piirmomendi/kiiruse graafik on seles 3.7. Graafikult on näha, et mootori kiiruse kasvades pöördemoment väheneb. See on põhjustatud sellest, et mähiste pideva ümberlülitamise tõttu indutseeritakse nendes vastuelektromotoorjõud. Seetõttu ei ole suurtel kiirustel võimalik saavutada enam nimivoolu.[19, lk 56]



Sele 3.7. Piirmomendi/kiiruse graafik [2, lk 42]

Selle graafiku järgi suudab mootor juhtida koormust 0.2 Nm kuni kiiruseni 500 sammu sekundis. 0.4 Nm juures suudaks mootor töötada kuni kiiruseni 200 sammuni sekundis ja lisaks oleksid probleemid kiirustel 20 ja 40 sammu sekundis.[2, lk 41]

Äkilised langused graafikul on põhjustatud sellest, et mähiste ümberpingestamine toimub sagedusel, mis läheb kokku rootori võnkumise omavõnkesagedusega [2, lk 49]. Seda efekti iseloomustab viirutatud joon seles 3.8. Kui printeri mõne mootori koormuse moment lõikab neid äkilisi langusi graafikul, siis tekib oht, et printer mingil hetkel jätab samme vahele ja prindib valesti. Sellisel juhul peaks kas omavõnkesagedusel töötamist vältima programmiliselt, kasutama mikrosamm-talitlust või lisama võnkumist summutava seadme mootorile.



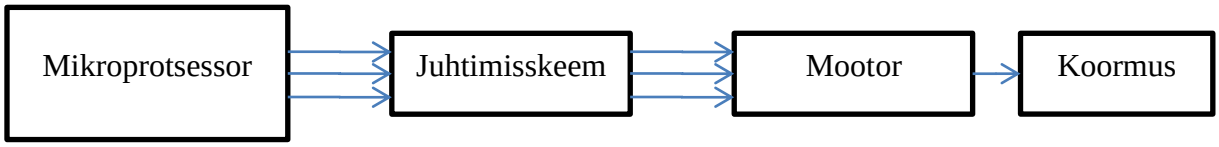
Sele 3.8. Rootori võnkumised kahel erineval kiirusel [2, lk 50]

Piirmomendi/kiiruse graafiku saab tavaliselt mootori andmelehelt. Samas see graafik sõltub väga palju samm-ajamist ja seega lõpliku graafiku peaks tekitama prototüübi katsetamise

käigus. Seda saab teha nii, et pannakse mootor käima teatud kiirusega ja hakatakse sellele koormust tasapisi lisama. Kui mootor seiskub, siis märgitakse see koormus üles antud kiiruse puhul. Seda peaks kordama veel paar korda antud kiirusega ja võtma keskmine saadud koormustest, et vähendada viga. Samamoodi peaks tegema teiste kiiruste puhul, kuni on saadud graafik.

3.1.4 Avatud juhtimisega ahel

Avatud juhtimisega ahela suurim eelis on see, et need on lihtsad ja odavad. Sele 3.9 näitab plokk-diagrammi tüüpilisest avatud juhtimisega süsteemist.

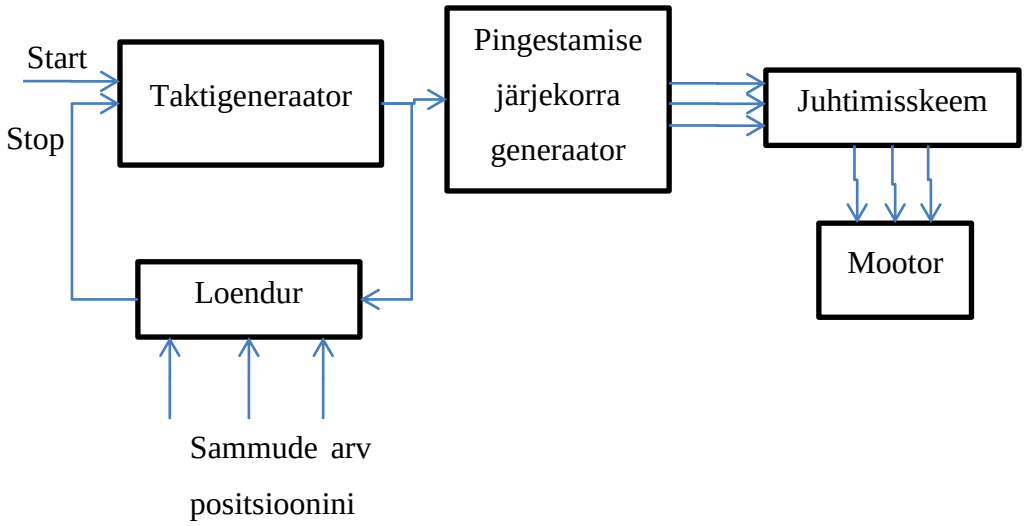


Sele 3.9. Avatud juhtimise ahela diagramm [2, lk 89]

Mikroprotsessor tekitab faaside juhtimiseks mõeldud signaale, mida võimendab juhtimisskeem enne, kui need jõuavad mootorini. Sellisele avatud süsteemile on erinevaid alternatiive, kus faaside juhtimiseks mõeldud signaale tekitatakse osaliselt riistvaras.[2, lk 89]

Avatud juhtahela puhul peab teadma, millised piirangud on juhtsignaalide ajastamisel. Mõned sellised piirangud tulenevad staatilisest olekust nagu on kirjeldatud peatükkides 3.1.2 ja 3.1.3, aga neile lisandub veel piirangud süsteemi dünaamilisest olekust.[2, lk 89]

Kõige lihtsam viis mootori juhtida oleks **konstantse kiirusega**. Sellise juhtimise diagramm on seles 3.10.



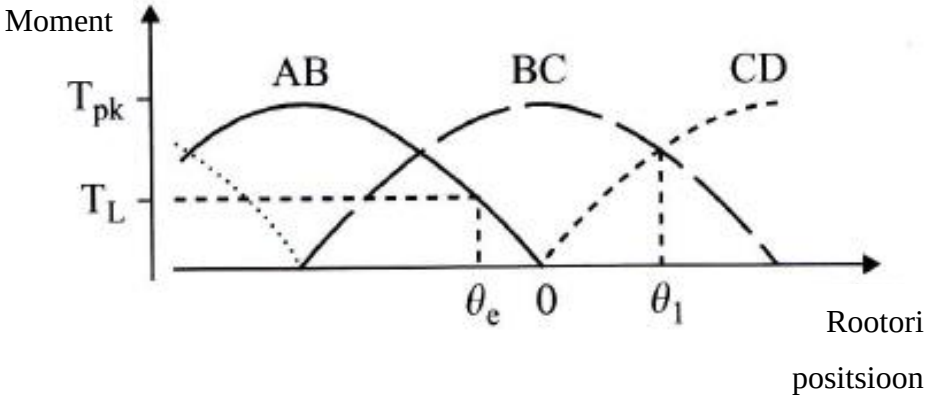
Sele 3.10. Konstantse kiirusega samm-mootori juhtimise diagramm [2, lk 90]

Mootori liikuma panemiseks tuleb kõigepealt *Start* signaal taktigeneraatorisse, mis saadab signaali pingestamise järjekorra generaatorisse, mille ülesanne on tekitada faaside juhtimiseks mõeldud signaale. Neid signaale võimendab juhtimisskeem, mis on ühenduses mootoriga. Taktigeneraatorist läheb signaal ka loendurisse, mis teab, mitu sammu peab tegema, ja loeb alla kuni nullini, peale mida annab taktigeneraatorile *Stop* signaali ning mootor peatatakse.[2, lk 90]

Kui taktigeneraator tekitab signaale liiga suure sagedusega, siis mootor ei suuda kiirendada koormusest ette nähtud kiiruseni ja süsteem kas lakkab töötamast või jätab samme vahele. Suurimat sagedust, millega koormatud samm-mootorit võib käivitada, et see ei jätaks samme vahele, on käivitussagedus. Samuti on veel olemas peatumissagedus, mis on analoogne peatumise korral. Konstantse sagedusega juhtimise korral peab taktsagedus olema väiksem peatumis- ja käivitussagedusest.[2, lk 91]

Üks võimalus käivitussageduse välja selgitamiseks on teha palju katseid nullist, aga seda saab ka ligikaudselt teada arvutuslikult. Oletades, et neljafaasiline mootor on paigal nii, et faasid A ja B on pingestatud nagu on näidatud seles 3.11. Sellisel juhul on staatiline viga antud valemiga (3.3). Esimene samm pingestab faasid B ja C ning staatiline pöördemoment positsioonis θ_e ületab koormusest tekkinud momenti. Seega mootor kiireneb positiivses suunas. Esimest sammu tehes peab mootor piisavalt kaugele liikuma, et süsteem oleks sünkroonis, kui toimub ümberpingestamine BC-st CD-ni. Siiski mootor ei pea liikuma tervet

sammu selle aja jooksul, sest väikse mahajäämise saab taastada järgnevate sammudega. Võib oletada, et rootor peab liikuma selle punktini, kus faaside momendid kattuvad, sest siis suudab mootor tekitada piisava pöördemomendi, et edasi kiirendada. Seles 3.11 on selleks punktiks θ_1.[2, lk 91]



Sele 3.11. Pöördemomendi/rootori positsiooni graafik neljafaasilise mootori puhul [2, lk 91]

Sellistes oludes keskmine pöördemoment, mida mootor tekitab liikudes positsioonist $\theta_e - \theta_1$, on:

$$T_M = \frac{1}{\theta_1 - \theta_e} \cdot \int_{\theta_e}^{\theta_1} -T_{PK} \cdot sin(p \cdot \theta - \frac{\pi}{2}) \cdot d\theta$$

$$= \frac{T_{PK}}{p \cdot (\theta_1 - \theta_e)} \cdot [\cos\left(p \cdot \theta_1 - \frac{\pi}{2}\right) - \cos\left(p \cdot \theta_e - \frac{\pi}{2}\right)] \quad \text{[2, lk 92]}, \quad (3.15)$$

$$= \frac{T_{PK}}{p \cdot (\theta_1 - \theta_e)} [\sin(p \cdot \theta_1) - \sin(p \cdot \theta_e)]$$

kus T_M – keskmine pöördemoment,

θ_1 – rootori positsioon pärast liikumist,

θ_e – rootori positsioon liikumise alguses.[2, lk 92]

Seda saab lihtsustada eeldades, et T_M on konstantne ja vaadates selet 3.11, siis see eeldus ilmselt ei tekita suuri vigu. T_{PK} erineb oma väärtuselt 30 %, mis tekitab suurima vea umbes 15 %. Seega võib kirjutada süsteemi inertsi kohta valemi:

$$T_M - T_L = J \cdot \frac{d^2\theta}{dt^2} \quad \text{[2, lk 92]} \quad (3.16)$$

kus T_L – koormuse poolt tekitatud pöördemoment,

J – süsteemi inerts.[2, lk 92]

Integreerides seda kaks korda aja järgi ja kasutades algtingimusi, kus $t = 0$, $\theta = \theta_e$ ja $d\theta/dt = 0$ saab kirjutada:

$$\theta = \frac{(T_M - T_L) \cdot t^2}{J} + \theta_e \quad \text{[2, lk 92]} \tag{3.17}$$

Järgmise sammu ajal on aeg t_p ja rootor positsioonis θ_1 ning seega:

$$\theta = \frac{(T_M - T_L) \cdot {t_p}^2}{J} + \theta_e => t_p = \sqrt{\frac{J \cdot (\theta_1 - \theta_e)}{T_M - T_L}} \quad \text{[2, lk 92],} \tag{3.18}$$

kus t_p – sammu tegemiseks kulunud aeg.

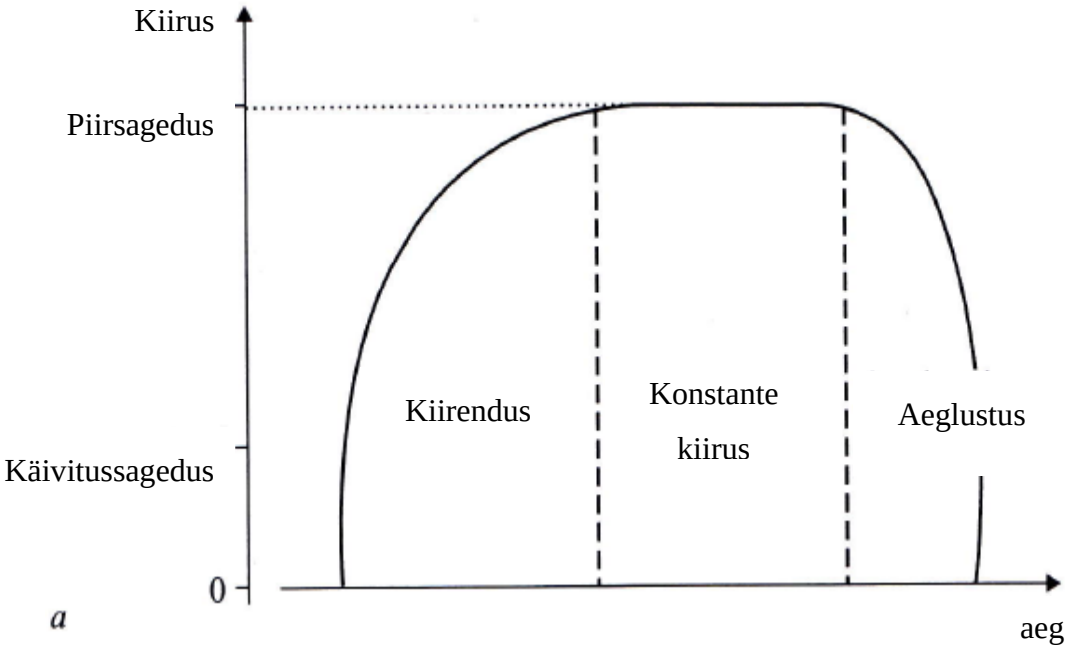
Siit tuleneb, et käivitussagedus on ligikaudu:

$$\mathrm{f} \approx \frac{1}{t_p} = \sqrt{\frac{T_M - T_L}{J \cdot (\theta_1 - \theta_e)}} \quad \text{[2, lk 92],} \tag{3.19}$$

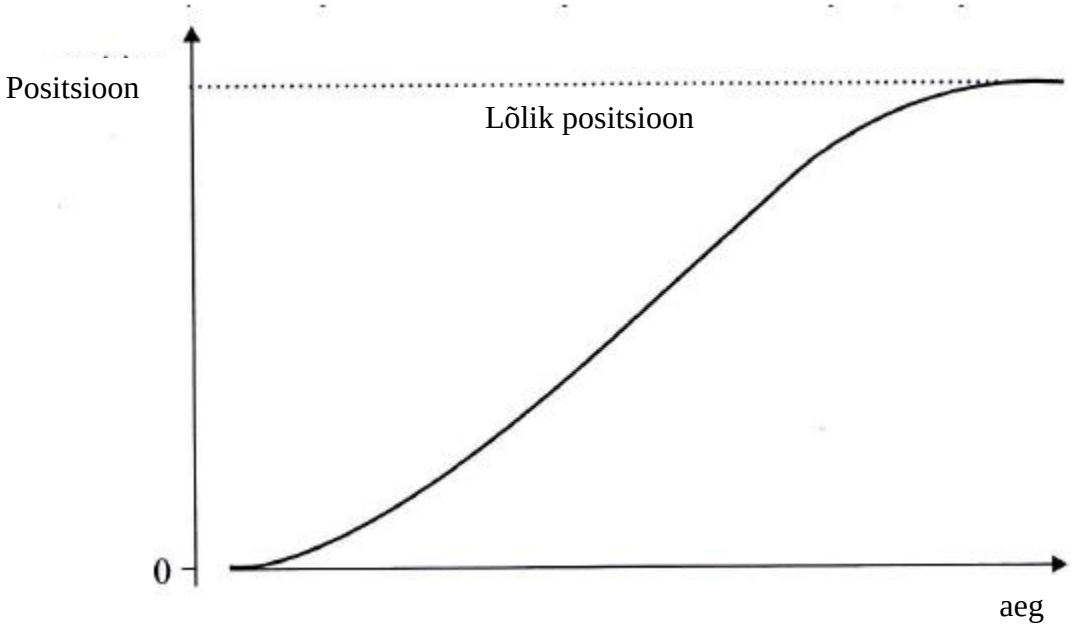
kus f – käivitussagedus.

Sellest valmist on näha, et käivitussagedus on suurem, kui mootor suudab tekitada suuremat pöördemomenti ja kui koormus on väiksem. Projektis kasutatavad haarad on üpris suured ja need arvatavasti tekitavad suure koormuse, kuid seda saab reaalselt hinnata alles katsetuste käigus. θ_1 saab mootori momendi/rootori positsiooni graafikust.

Tavaliselt on mootori töötamise sagedus, millega see staatilises olekus samme vahele ei jäta (piirsagedus) palju suurem käivitussagedusest. Juhul kui tuleb välja, et printer peab liikuma käivitussagedusest kiiremini, siis on vaja, et süsteem jätkaks kiirendamist pärast esimest sammu. Samamoodi peaks lõpp-positsioonile lähenedes aeglustama peatumissageduseni. Funktsiooni mootori kiirusest/positsioonist ja ajast kiirendamisel ja aeglustamisel nimetatakse kiirendus- ja pidurduskõveraks. Samuti võib neid kutsuda **kiiruse ja liikumise profiilideks**. Tüüpiline sellise kiiruse profiili graafik on selel 3.12 ja liikumise profiil on selel 3.13. Graafikult on veel näha, et tavaliselt on kiirendus aeglasem aeglustusest.[2, lk 94]



Sele 3.12. Näide kiiruse profiili graafikust [2, lk 95]



Sele 3.13. Näide liikumise profiili graafikust [2, lk 95]

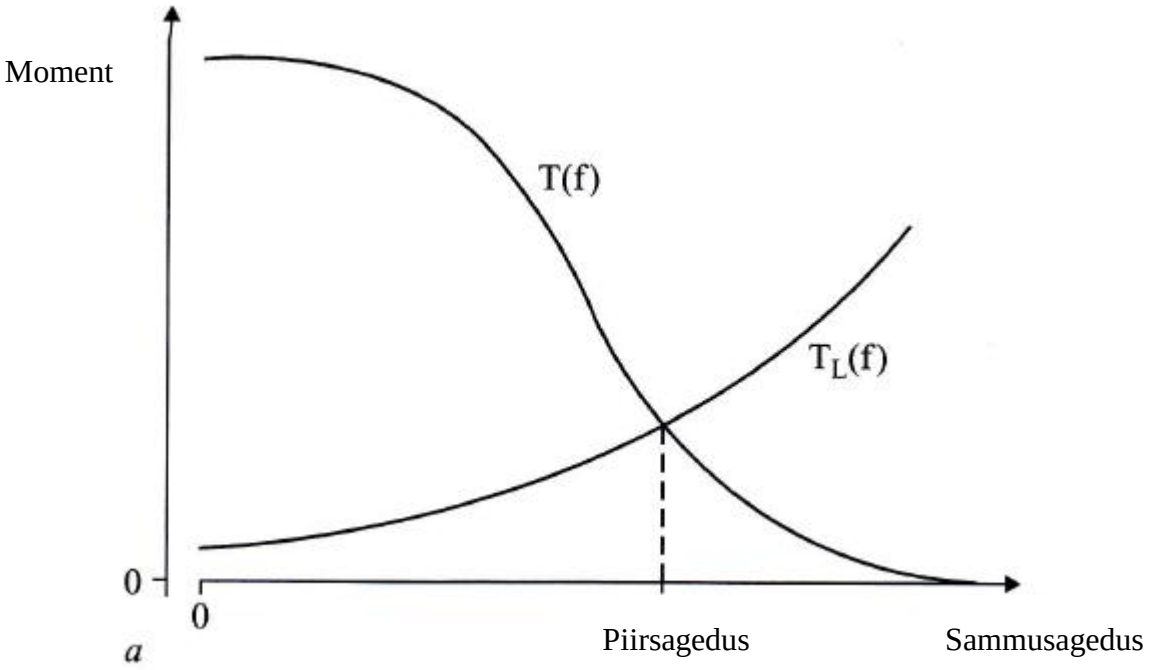
Seega peab programm tekitama kiiruse profiili, mis on võimalikult sarnane sellisele graafikule. Üks võimalus seda luua on uurides piirmomendi ja koormusest tekkivat momenti. Näide seda iseloomustavast graafikust on seles 3.15. Sammusagedusel *f* on piirmoment *T(f)* ja koormuse moment $T_L(f)$. Mootori kiirendamisel maksimaalse võimaliku kiirusega peab piirmoment olema suurim kõikidel kiirustel. See moment on suurem koormuse momendist ja seega saab kirjutada:

$$T(f) = T_L(f) + J \cdot \frac{d^2\theta}{dt^2} \quad \text{[2, lk 95]}, \tag{3.20}$$

kus f – sammusagedus,

$T(f)$ – piirmoment sammusagedusel f,

$T_L(f)$ – koormuse moment sammusagedusel f.[2, lk 95]



Sele 3.15. Näide piirmomendi *T(f)* ja koormuse momendi $T_L(f)$ graafikutest [2, lk 96]

Mootoril, millel on n faasi ja p rootori hammast, on sammu pikkus $2\pi f/np$ ja seega sammusagedus on seotud rootori kiirusega [2, lk 96]:

$$\frac{d\theta}{dt} = \frac{2 \cdot \pi \cdot f}{n \cdot p} \quad \text{[2, lk 96]} \tag{3.21}$$

Asendades valemi 3.21 valemisse 3.20 saab kirjutada:

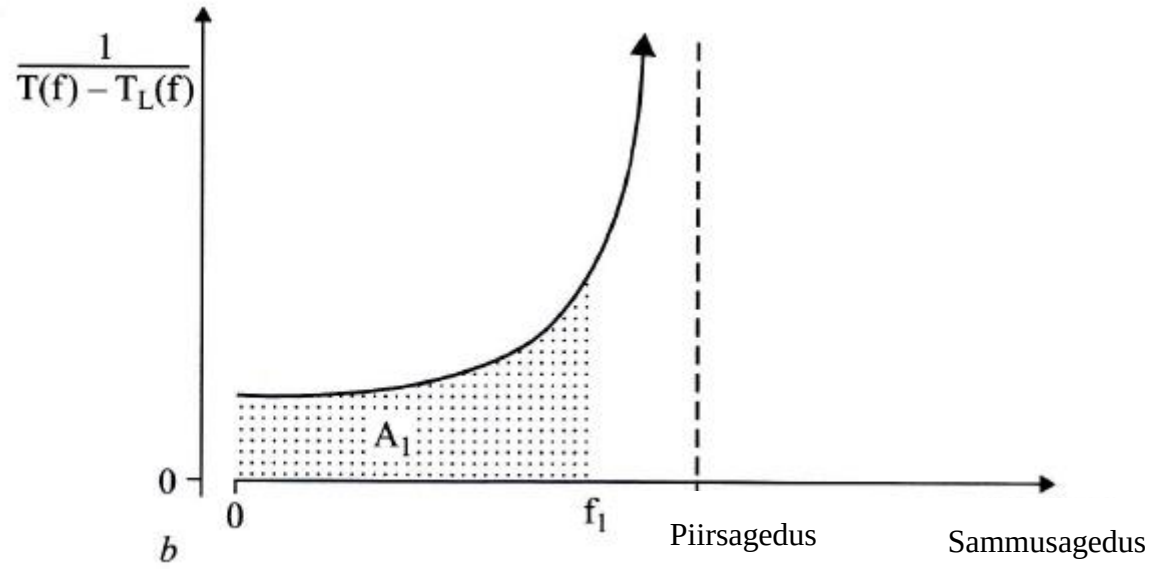
$$T(f) = T_L(f) + J \cdot \frac{2 \cdot \pi}{n \cdot p} \cdot \frac{df}{dt} => \frac{df}{dt} = [T(f) - T_L(f)] \cdot \frac{n \cdot p}{2 \cdot \pi \cdot J} \quad \text{[2, lk 96]} \tag{3.22}$$

Seda valemit saab integreerida, et saada *t*, mis näitab kui palju kulub aega, et saavutada sammusagedus *f*, kui mootor kiirendab nullist [2, lk 96].

$$\frac{n \cdot p}{2 \cdot \pi \cdot J} \int_0^t dt = \frac{n \cdot p \cdot t}{2 \cdot \pi \cdot J} = \int_0^f \frac{df}{T(f) - T_L(f)} \quad \text{[2, lk 96]}, \tag{3.23}$$

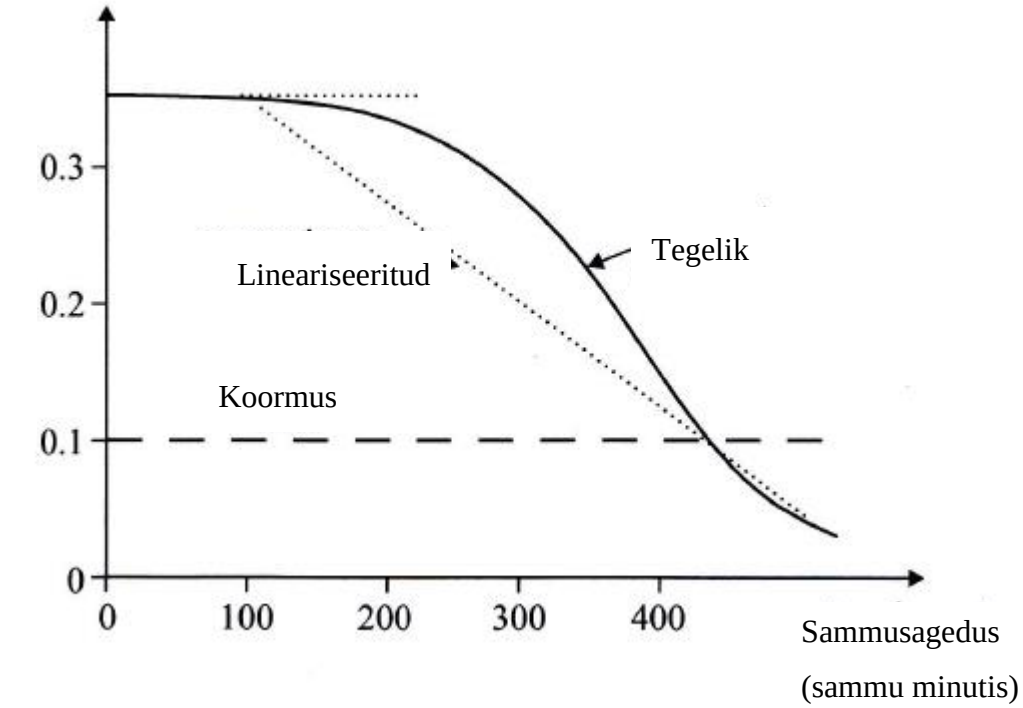
Tavaliselt peab seda integreerimist tegema graafiliselt, sest funktsioonid $T(f)\ ja\ T_L(f)$ on mitteanalüütilised [2, lk 97]. Seles 3.16 on näha värvitud ala A_1, mis on see leitud integraal. Seega aeg t_1, mida on vaja, et saavutada sagedus f_1, on leitav valemiga:

$$t_1 = \frac{2 \cdot \pi \cdot J \cdot A_1}{n \cdot p} \quad \text{[2, lk 97]} \tag{3.24}$$



Sele 3.16. Pindala leidmine funktsioonides $T(f)\ ja\ T_L(f)$ [2, lk 96]

Terve kiirenduskõvera saab leida, kui seda protsessi korrata mitu korda kuni mootori piirsageduseni. Selleks, et programm suudaks neid kõveraid leida, võib funktsioone $T(f)\ ja\ T_L(f)$ lineariseerida nagu on seles 3.17. Neid funktsioone ei ole võimeline programm leidma, kui see teab ainult järgmist sammu ja seega peab programm suutma analüüsida mitu sammu ette.[2, lk 97]



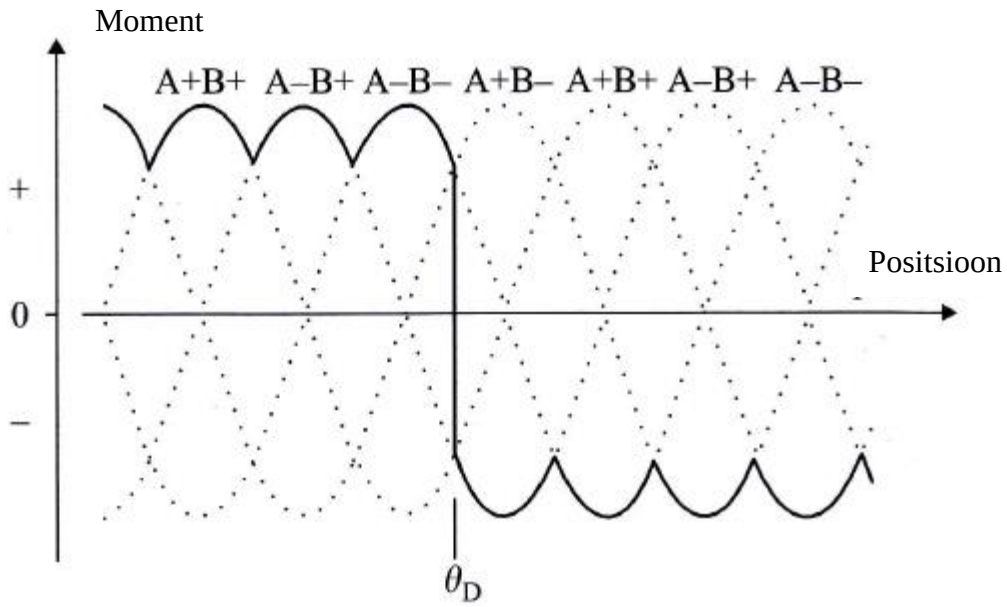
Sele 3.17. Piirmomendi lineariseerimine [2, lk 96]

Aeglustades peaks kasutama analoogselt kasutatud valemit:

$$\frac{n \cdot p}{2 \cdot \pi \cdot J} \int_0^t dt = \frac{n \cdot p \cdot t}{2 \cdot \pi \cdot J} = \int_{f_1}^{f} \frac{df}{T_B(f) + T_L(f)} \quad \text{[2, lk 99],} \tag{3.25}$$

kus $T_B(f)$ - aeglustamisel tekkinud moment funktsioonina sammusagedusest.

Üleminek piirkiiruselt, kus mootor tekitab positiivset momenti, aeglustuseni, kus mootor tekitab negatiivset momenti, on saavutatav sellega, kui n-ö hüpatakse pingestamise järjekorras tagasi [2, lk 99 - 100]. Sele 3.18 näitab sellist üleminekut bipolaarsel samm-mootoril.



Sele 3.18. Mähiste pingestamise järjekord aeglustama hakkamisel [2, lk 100]

Sellelt selelt on näha, et positsiooni θ_D juures ei ole järgmine samm A+B-, vaid on A-B+. Seega on sammude järjekord nihkunud kahe sammu võrra ja sammude loendurit peab vastavalt muutma [2, lk 100].

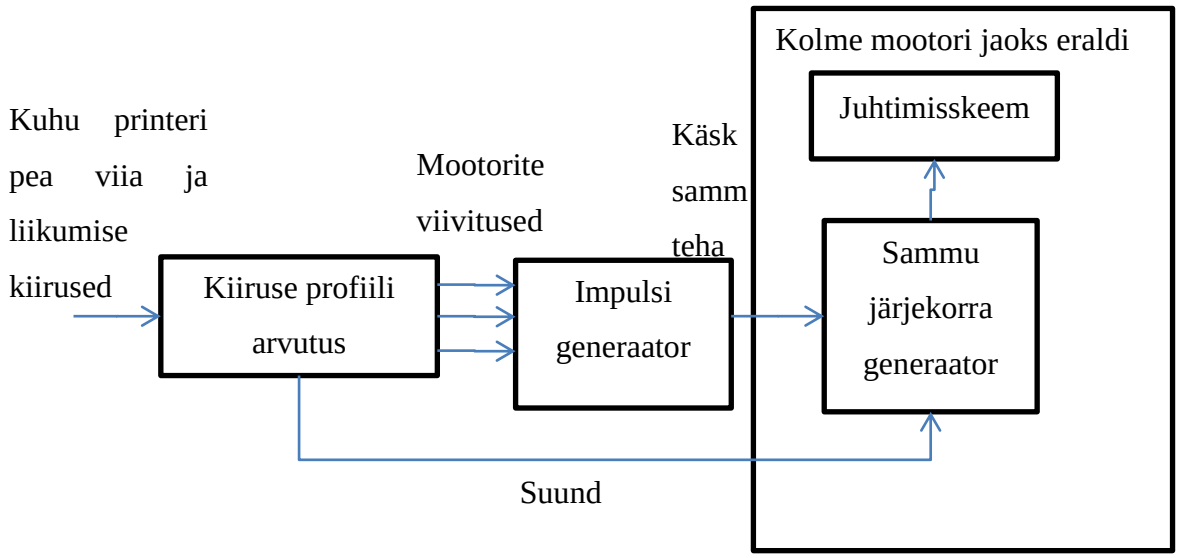
Positsiooni θ_D juures aeglustust on võimalik veel tekitada sellega, et mootorit hoitakse olekus A-B- kolm korda pikemalt, sest selle ajaga on rootor piisavalt edasi liikunud, et tekitada negatiivset momenti. Viimast meetodit kasutades aeglustub mootor pikemat aega, aga sammude lugemise poole pealt on see lihtsam.[2, lk 100]

Üks võimalus kiiruseprofiili loomiseks on veel kasutades ära täpset aega, millal järgmise sammu peab tegema. Selleks on vaja positsiooni/aja graafikut, mille näide on seles 3.13, mis saadakse, kui integreeritakse kiiruse profiili.[2, lk 100]

3.1.5 Avatud ahela realiseerimine 3D printeris

Avatud ahela realiseerimise võib jagada erinevateks mooduliteks, mis kokku moodustavad samm-mootorite juhtimise süsteemi. 3D printeri puhul sisend sellesse süsteemi on asukohad, kuhu mootorid peavad printeri pea viima, ja pea liikumise kiirus. Samm-mootorite juhtimiseks on vaja tekitada impulsse, millal järgmine samm teha ja nende impulsside järgi õiges järjekorras mootori mähiseid pingestada. Samuti peab süsteemis olema moodul, mis

arvutab välja iga mootori jaoks eraldi kiiruse profiili. Samm-mootoreid on kolm ja seega nende juhtimiseks mõeldud süsteemi saab kujutada nii, nagu on seles 3.19.



Sele 3.19. 3D printeri mootorite juhtimise diagramm

Neid mooduleid (peale juhtimisskeemi) saab rakendada kas mikrokontrolleriga koodis või riistvaraliselt. Selles töös on eesmärgiks pakkuda programmiline lahendus neile ja seega riistvaralist võimalust ei uurita. Internetist on võimalik saada vabavarana valmis kirjutatud koodi nende moodulite jaoks.

Kiiruse profiili arvutuse mooduli eesmärgiks on analüüsida asukohti, kuhu printeri pea peab liigutama, ja arvutama välja iga mootori jaoks aja, millal järgmist sammu teha. Mootorite kiiruste arvutamisel peab printeri pea kiirendama võimalikult ruttu G-koodist loetud kiiruseni v.

$$v = \sqrt{v_x^2 + v_y^2 + v_z^2}, \tag{3.26}$$

kus v_x – X telge liigutava mootori kiirus,

v_y – Y telge liigutava mootori kiirus,

v_z – Z telge liigutava mootori kiirus,

v – printeri pea liikumise kiirus.

Kui ette antud kiirus on suurem, kui mootorite maksimaalsed kiirused, siis peab moodul andma veateate ning toimima vastavalt sellele, kuidas on veatöötluse moodulis kirjutatud. Maksimaalset kiirust saab leida nii, nagu on kirjas peatükis 3.1.3.

Süsteemi väljundiks on kolm lingitud nimistut. Selles nimekirjas on number, mis näitab mitu tsüklit peab programm viivitama, et see samm teha. Samuti on seal muutuja, kus on salvestatud, mitu tsüklit on viivitatud ja muutuja, kus on kirjas, mis suunas mootor peab liikuma. Suuna märkimiseks saaks näiteks kasutada korralist loetelu (*enumeration*). Töö käigus vigade leidmiseks võiks olla veel lisaks muutuja, kus on kirjas, kas antud samm on tehtud. Tabelis 3.2 on kujutatud mooduli väljundiks olevat lingitud nimistu näidist, milles arvutatakse ette 9 sammu.

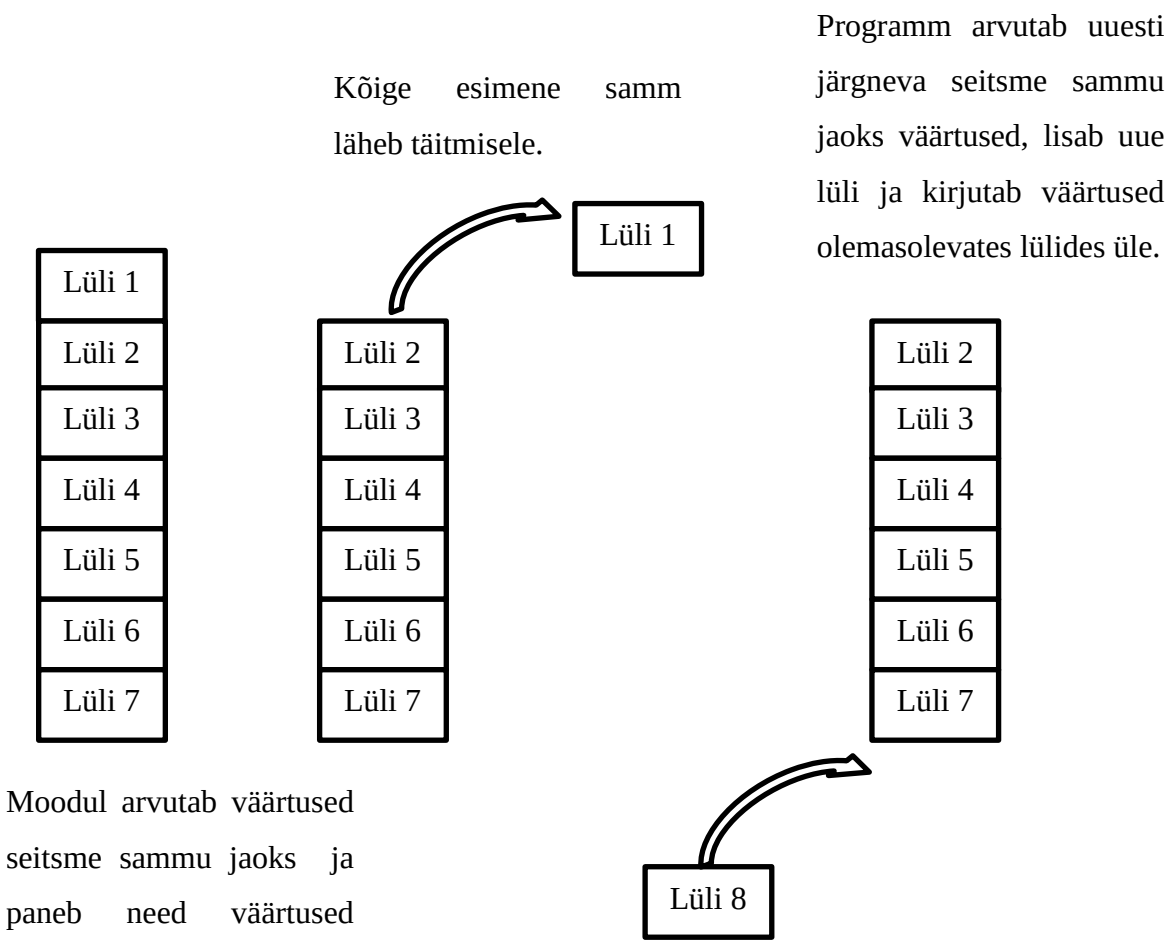
Tabel 3.2. Lingitud nimistu, kus on kirjas, kuidas mootor peaks samme sooritama

Lüli nr.	Viivitus	Sammu teostatud	Suund	Samm tehtud?	Viit lülile
1	0	0	Edasi	0	2
2	25	0	Edasi	0	3
3	21	0	Edasi	0	4
4	18	0	Edasi	0	5
5	16	0	Tagasi	0	6
6	14	0	Tagasi	0	7
7	12	0	Tagasi	0	8
8	11	0	Tagasi	0	9
9	10	0	Tagasi	0	

Konstantse kiirusega juhtimine on lihtsam, aga *Sliceri* poolt genereeritud G-koodis on printeri pea muutuva kiirusega. Kõigepealt võib mootorid käima panna konstantse kiirusega, et kätte saada piirmomendi/kiiruse graafiku, aga lõpp lahenduses peab arvatavasti võimaldama muutuva kiirusega liikumist. Konstantse kiirusega juhtimisest on kirjutatud peatükis 3.1.4.

Optimaalsemate kiiruste saavutamiseks peaks arvutama mootorite kiirendused ja aeglustused välja vastavalt peatükile 3.1.4. Kõige ligilähedasema kiiruse profiili ideaalile saab siis, kui arvutada see välja arvestades kõiki asukohti, kuhu printeri pea peab viima. Selle arvutamisega ei saa mikroprotsessor hakkama, sest neid asukohti on liiga palju. Järelikult peab programm kõigepealt välja arvutama kiiruse profiili ainult teatud arvu sammude jaoks. Kui mõni samm on täidetud, siis peab programm uue sammuga arvestama ja kõik eelnevad täitmata sammud selle järgi üle arvutama. Seles 3.20 on kujutatud mooduli väljundiks olevat lingitut

nimistukirja lülisid ja kuidas moodul käitub nendega. Kui esialgu kiirendusi ja aeglustusi ei arvutada, siis ei pea ette arvutama samme ja lingitud nimistus võib olla ainult üks väärtus.



Sele 3.20. Sammu informatsiooni sisaldavate lülide arvutamine ja täitmine

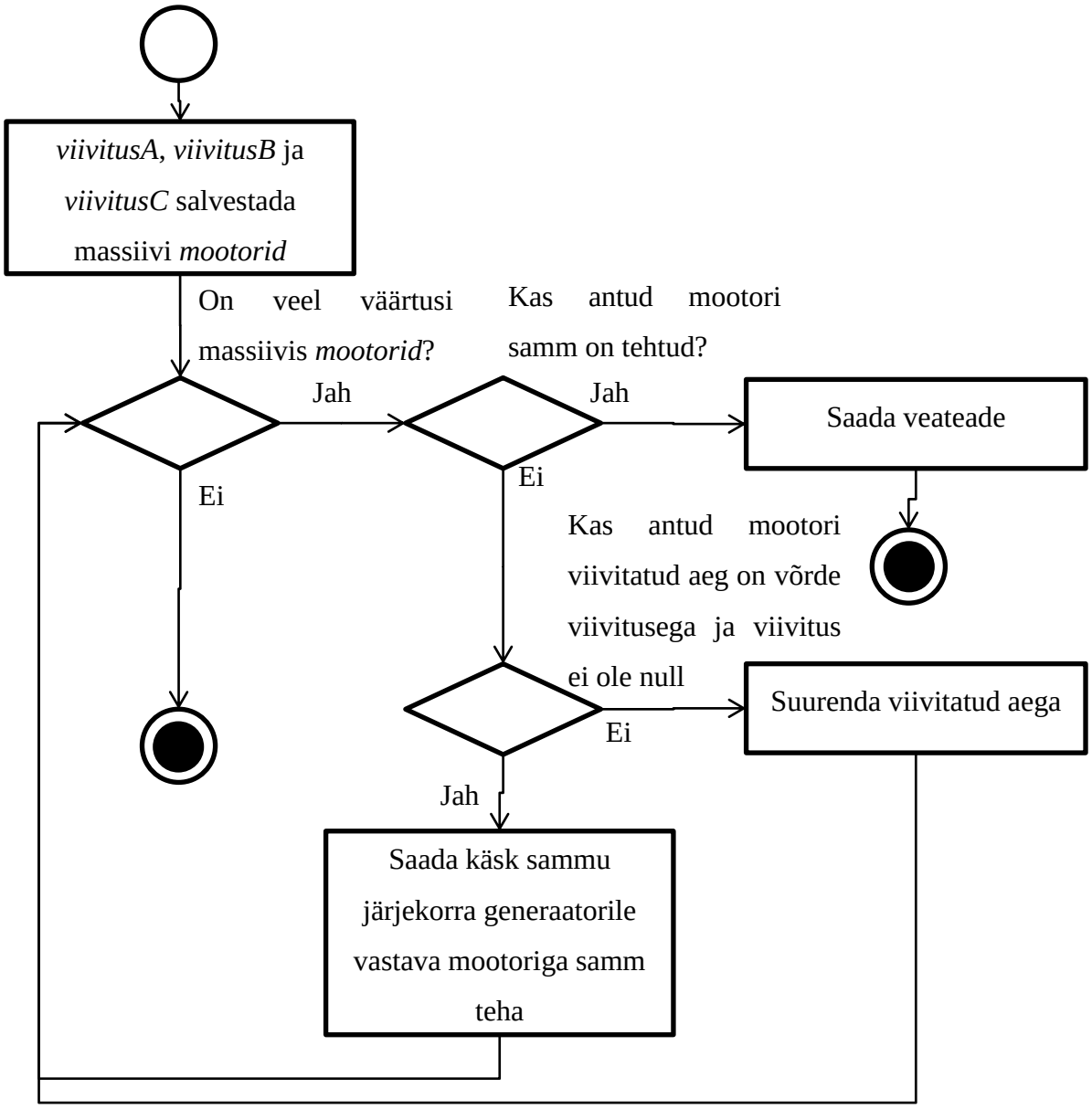
Katsetuste käigus võib ilmneda, et printeri pea võtab nurki liiga aeglaselt. Selle lahenduseks võib nurki ümardada.

Arvestama peab juhtimise talitlusega, millest on kirjas peatükis 3.1.2, sest see määrab ära, kui palju mootor liigub ühte sammu sooritades. Mikrosamm-talitluse puhul peab arvestama sellega, et iga lüli jaoks arvutatud sammu pikkus on väiksem.

Impulsi generaatori eesmärgiks on väljastada signaale varem välja arvutatud sammusagedusega. Kui süsteem koosneks ühest mootorist, siis selle sisendiks oleks täisarv, mis näitab, mitme tsükli pärast peab mootor sammu tegema. 3D printeril on haarade juhtimiseks kolm mootorit ja seega on sisendeid ka kolm täisarvu.

Kõige mõistlikum on välja kutsuda taimeri abil katkestust aja tagant, mis on mootori piirsageduse kordne. Katkestuse väljakutsumise sagedus peab olema piisavalt väike, et protsessor jõuaks kõiki mootoreid juhtida. Seles 3.21 on voodiagramm võimalikust programmilisest lahendusest ühe mootori juhtimiseks. Kiiruse profiili arvutava mooduli väljund on kolm lingitud nimistut, mille esimestes lülides on impulsi generaatori sisend. Nende lülide aadressid on salvestatud globaalses viit tüüpi muutujas. Selles näites on viit tüüpi muutujateks vastavalt mootorite A, B ja C puhul *viivitusA*, *viivitusB* ja *viivitusC*. Nendes lülides on kaks muutujat. Üks näitab, mitu tsüklit peab viivitama, et see samm teha, ja teine näitab, kui palju on juba viivitatud. Samuti on seal viit tüüpi muutuja, kus on kirjas järgmise lüli aadress.

Programm kõigepealt salvestab muutujad *viivitusA*, *viivitusB* ja *viivitusC* massiivi, mille nimi on *mootorid*. Seejärel analüüsib programm kõigi kolme mootori viivitusi ükshaaval kordlausega. Kui analüüsitava mootori puhul on see samm juba tehtud, siis programm saadab veateate moodulisse *Veatöötlus*. Kui sammu ei ole tehtud ja piisavalt aega on möödas järgmise sammu tegemiseks ja viivituse väärtus ei ole null, mis tähendaks, et selle mootoriga sammu ei ole vaja sooritada, siis programm saadab käsu sammu järjekorra generaatorile vastava mootoriga samm sooritada. Kui piisavalt aega ei ole möödas, siis viivitatud aja muutujat suurendatakse. Seejärel analüüsitakse järgmist mootorit. Selle lahenduse probleemiks võib kujuneda see, et tekib väike nihe funktsiooni väljakutsumise ja sammu signaali saatmise vahel. Lisaks ei hakka mootorid samal ajal liikuma. Teine võimalus oleks kasutada FPGA-d (*field-programmable gate array*), mis võimaldab mootorid korraga liikuma panna. See lahendus oleks palju kallim ja kuna samm-mootori pingestamise sagedus on kordades väiksem mikrokontrolleri taktsagedusest, siis sobib ka esimene lahendus.



Sele 3.21. Voodiagramm sammu täitmise ajastamisest

Sammu järjekorra generaatori ülesandeks on mootori mähiseid õiges järjekorras pingestada. Selle väljundit peab võimendama elektriskeemiga. Sisend sellele moodulile on käsud, millal järgmine samm teha ning sammu suund. Samm-mootorite pingestamise järjekorra põhimõttest on rohkem kirjas peatükis 3.1.1. Pingestama peab vastavalt valitud juhtimise viisi talitlusest, millest on kirjas peatükis 3.1.2.

Kui mootoreid juhtida muutuva kiirusega, siis peab arvestama sellega, et üleminekul aeglustumisele peab sammude järjekorda muutma. Sellest on pikemalt kirjas peatüki 3.1.4 lõpu poole.

3.2 G-koodi juhtmoodulisse edastamine

G-koodi juhtmoodulisse saaks kas läbi õhu, juhtme või mälukaardiga. Läbi õhu saatmise eelis on selles, et seadme ja kasutaja vahel saab toimuda pidev andmevahetus. Samuti võivad kasutaja või seade vabalt liikuda ilma, et ühendus katkeks. Tänapäeva mikrokontrolleritel ei ole läbi õhu saatmiseks mõeldud moodulit tavaliselt rakendatud ja seega selle peaks eraldi juurde lisama.

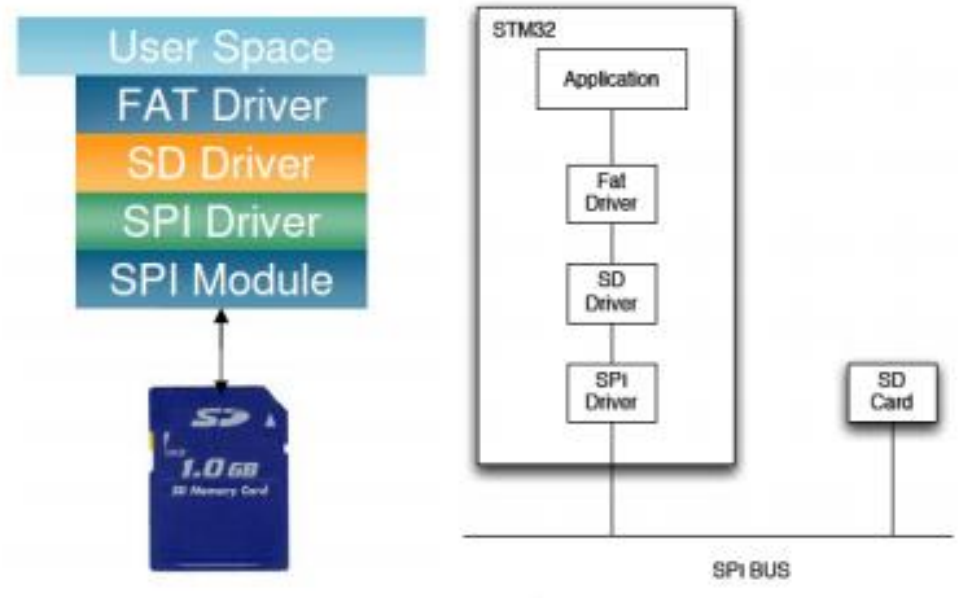
Juhtmega info saatmine võimaldab samuti pidevat andmevahetust. Mikrokontrolleritel on tavaliselt moodulid selle jaoks olemas (UART - *universal asynchronous receiver/transmitter*, USB – *universal serial bus*).

Mälukaardi puhul saaks kasutada G-koodi faile, mis on suuremad, kui on juhtmoodulil mälu. Selle rakendamise puhul on väiksem oht info kadumisel ja ei pea realiseerima keerukat suhtlusprotokolli. Samuti on paljudel mikrokontrolleritel mälukaardi jaoks pesa olemas.

Antud 3D printer hakkab arvatavasti suuri esemeid välja printima ja seega nende tegemiseks võib minna päevi. Seega pidev andmevahetus oleks tülikas. Mälukaarti kasutades saab eelnevalt G-kood valmis teha ja see kaardilt otse printerisse laadida ilma, et peaks vahepeal arvuti käima panema ja selle printeriga ühendama. See tundub suure printeri puhul kõige mõistlikum lahendus ja töö praktilises osas on kasutatud G-koodi edastamiseks mälukaarti.

3.2.1 Mälukaardi draiver

Mälukaardilt lugemiseks on vaja draiverit, mis suudaks kaardiga suhelda ja seal infot kätte saada. Mälukaarte on kahte tüüpi: SD (S*ecure Digital*) ja MMC (*MultiMediaCard*) kaardid. SD kaardid on edasiarendus MMC kaartidest ja arvatavasti printeriga hakatakse neid kasutama.[5]

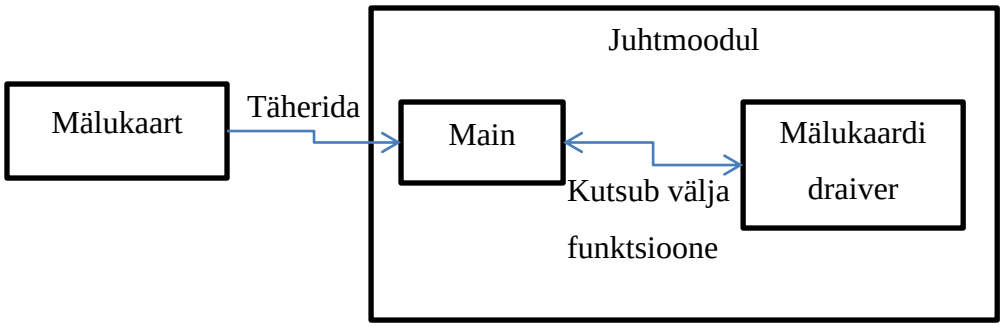


Sele 3.22. SD kaardi tarkvara kihid [17]

Mälukaariga suhtlemiseks on kaks liidest: SD ja SPI (*Serial Peripheral Interface*). Vaikimisi kasutavad mälukaardid SD ühendust, aga neid saab panna SPI liidest kasutama. Suurem osa mikrokontrollereid ei toeta SD ühendust ja seega on palju mõistlikum kasutada SPI liidest, mis on samuti palju lihtsam. Seega juhtmooduli programm peab olema võimeline panema mälukaarti SPI liidest kasutama.[17]

Lisaks suhtlemise protokollile on vaja failihaldus liidest, mis võimaldab kaardilt infot lugeda ja sinna seda kirjutada. Populaarseim neid on FAT (*File Allocation Table*) failisüsteem, mis omakorda jaguneb FAT12, FAT16 ja FAT32-ks.[17]

Seega 3D printeri programmi mälukaardi draiver peab toetama SPI ühendust ja ühte või mitut FAT failisüsteemi. Internetis on nendele nõuete vastav moodul, mille on teinud ChaN ja mis on vabavarana saadaval. See on saadaval internetileheküljel, millele viitab viide [5]. See on kirjutatud ANSI (*American National Standards Institute*) C-s ja on eraldiseisev moodul süsteemi sisend/väljund kihist [5].



Sele 3.23. Mälukaardiga suhtlemise diagramm

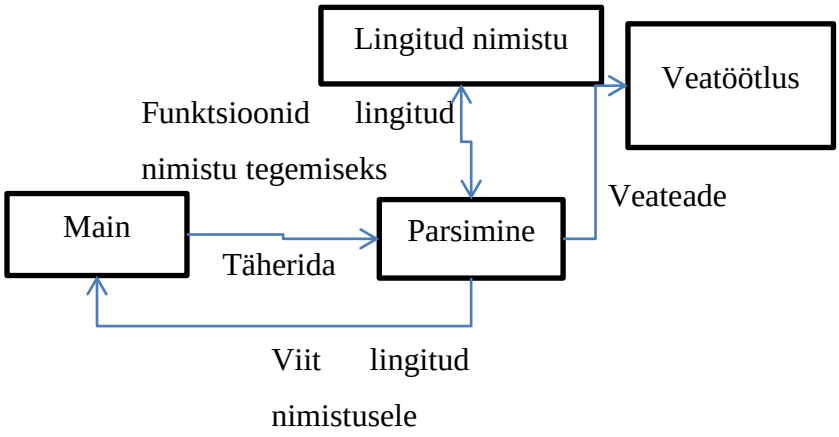
3.2.2 RS274 parsimine

RS274 kood on jaotatud ridadeks, kus on üks või mitu „sõna“, mis koosneb tähest ja numbrist [10]. Sellest lähtuvalt kõige mõistlikum on parsida täherida lingitud nimistusse (*linked list)*. See on mälustruktuur, mis koosneb lülidest. Igas lülis on informatsioon ja indikaator järgmise lüli mälupesale. Järelikult peab juhtmoodulis olema moodul, mis võtab sisendiks mälukaardilt loetud täherea ja parsib selle lingitud nimistu.[16]

```
----------        ----------
- Data   -        - Data   -
----------        ----------
- Pointer- - - -> - Pointer-
----------        ----------
```

Sele 3.24. lingitud nimistu mälus [16]

Parsimise ajal käib programm terve sisendkoodi läbi ja kontrollib, kas see vastab RS274 standardile.



Sele 3.25. Diagramm moodulitest, mis parsivad RS274 koodi

3.3 Muud sisendid ja väljundid

3.3.1 Küttega alusplaat

Programm peab suutma 3D printeri küttega alusplaati juhtida. See peab viima alusplaadi teatud temperatuurile ja hoidma seda konstantsena. Hetkel projektis alusplaati ei ole rakendatud ja seetõttu ei ole kindel, milline on tagasiside. Temperatuuri vajalikul tasemel hoidmiseks võib kasutada PID-i (*proportional-integral-derivative*) või lihtsalt mingis temperatuuri vahemikus sisse välja lülitamist [13, lk 29 - 30]. PID-i kasutamisega oleks alusplaadi temperatuur palju täpsemalt paigas, aga selle täide viimine on keerukam. Samuti nõuaks see suuremat protsessori ressurssi. Teine võimalus on, et kui näiteks tahetakse temperatuuri hoida 100 °C juures, siis lülitada alusplaadi küte välja, kui on saavutatud 102 °C ja uuesti sisse, kui on saavutatud 98 °C. Selle teostamine on palju lihtsam ja kuna alusplaat jahtub aeglaselt, siis see säästaks väga palju protsessori ressurssi.

Programmis peab seega mingi aja tagant kontrollima, mis on alusplaadi temperatuur. Seda võib teha väga madala sagedusega, sest alusplaat jahtub aeglaselt võrreldes protsessori kiirusega. Siiski läheb selle jaoks programmis vaja taimerit ja kuna nende arv on mikrokontrolleril piiratud, siis peaks tegema ühe süsteemi taimeri. Sellisel juhul saaks teha nii, et iga mingi aja tagant suurendatakse kindlat muutujat ja kontrollitakse, kas mõnda protsessi peab välja kutsuma vastavalt sellele, kui suur on selle muutuja väärtus. Vajadusel peab termistori poolt antavat tagasisidet filtreerima.

3.3.2 Piirlülitid

Projektis kasutatavatel 3D printeri telgedel on mõlemal poolel piirlülitid. Need saadavad välja signaali, kui printeri pea on liikunud telje ühte äärde. Neid signaale saab ära kasutada tagasisidena printeri pea asukohast. Juhtmoodulil on vaja teada printeri pea algset positsiooni, et arvutada välja, kui palju peab mootoreid liigutama. Seega juhtmoodul peab saama infot vähemalt ühe lõpplüliti kohta igast haarast, et täpselt positsioneerida printeri pea algasendisse.

Printeri töö käigus võib tulla olukordi, kui printeri pea võib mõne eksimuse tõttu üritada liikuda telgedest välja. Selle vältimiseks peavad piirlülitid saatma juhtmoodulile signaali.

Juhtmoodul saab seejärel näiteks mootorid seisma jätta. Piirlülitid võiksid saata signaali mikrokontrolleri sellisesse sisendisse, mis kutsuks välja katkestuse programmis. Juhul kui iga lüliti jaoks ei jätku selliseid sisendeid, siis võiks kasvõi üks katkestusega sisend olla, mida kutsutakse välja, kui mõni lülititest on saatnud signaali. Lülitid võiksid siiski olla kõik eraldi mõne juhtmooduli sisendiga ühendatud ja kui katkestusega sisend signaali saab, kontrollib see kõiki teisi sisendeid, et teada, milline lüliti signaali saatis.

Seega peab programmis kasutusele võtma mooduli piirlülitite jaoks, mis suhtleb moodulitega Main ja Veatöötlus.

3.3.3 Printeri pea

Juhtmoodul peab juhtima printeri pea samm-mootorit ja kütteelementi. Selle lihtsamaks tegemiseks on projektis kasutusel elektroonika plaat. Sellel plaadil on samm-mootorite juhtimiseks mikroskeem A3979 [1]. Plaadi elektroonikaskeem ei ole saadaval, aga selle 3D printeriga varem töötanud tudengid uurisid plaadi sisend-väljund viikusid ja koostasid tabeli 3.3.[13, lk 17]

Tabel 3.3. Printeri pea elektroonika plaadi sisend-väljund viigud [13, lk 17]

Sisendviigu number pealtvaadates alates paremalt	Viigu eesmärk
1	Samm-mootori suund
2	Samm-mootori sammu signaal
3	GND
4	GND
5	Loogika toitepinge +5V
6	Samm-mootorite toitepinge +12V
7	Samm-mootorite toitepinge +12V
8	Samm-mootori lubamissignaal
9	Termistori signaal
10	Kütteelemendi juhtsignaal

Sellest tabelilt on näha, et samm-mootori juhtimiseks peab andma teisele viigule sammu signaale, mida saab tekitada impulsi generaator (sele 3.19) või teha eraldi süsteem. Kõige lihtsam oleks seda teha kasutades ära impusi generaatori funktsiooni, mida kujutab sele 3.21. Sellesse funktsiooni peaks lisama ühe mootori juurde ja kindla viivitusega sammusignaale edasi saatma. Arvatavasti printeri pead juhtiva samm-mootori kiirus on konstantne ja seega võib viivituste aeg olla ka konstantne.

Samm-mootori suunda ei ole arvatavasti vaja muuta ja seega võib esimese viigu kas kõrgeks või madalaks panna vastaval sellele, kummas suunas peab mootor pöörlema. Juhtmoodul peaks samm-mootoril liikumist lubama ainult siis, kui on tahetud objekti printida. Seda saab teha kaheksanda viigu abil. Samuti saab seda sisendit kasutada mootori seiskamiseks, kui tekib ohuolukord. Juhtmoodul peab veel saatma signaale printeri pea kütteelemendile kümnenda viigu abil ja saama infot temperatuuri kohta viigust üheksa. Seega printeri pea juhtimiseks olev moodul suhtleb moodulitega Main, Impulsi generaator ja Veatöötlus.

4. LÕPLIK LAHENDUS

Sele 4.1 näitab lõpliku lahenduse diagrammi. Igat nendest moodulitest võib täitmise käigus veel jaotada väiksemateks mooduliteks. Peatükis kolm on kirjas kuidas neid mooduleid realiseerida.

Main moodulis asub funktsioon *main*, kust hakkab programm peale. Selle mooduli eesmärk on juhtida kõikide teiste programmi osade tööd. See kõigepealt seab süsteemi tööks valmis. Seejärel see loeb sisse G-koodi kasutades selleks mälukaarti ja moodulit Mälukaardi draiver.

Sisse loetud kood on täherida ja sellega edasi tegeleb moodul Parsimine. See moodul parsib täherea lingitud nimistusse, mille iga lüli on üks G-koodi rida.. Samal ajal kontrollib, kas antud kood vastab NIST standardile [10]. Vea leidmisel kutsutakse välja funktsioon moodulist Veatöötlus. Lingitud nimistu tegemise ja sellega suhtlemise lihtsustamiseks on moodul Lingitud nimistu.

Pärast koodi parsimist hakkab moodul Main lingitud nimistust ükshaaval käsklusi võtma ja neid täitma. Kui tuleb käsk alusplaat teatud temperatuurile viia, siis seda teeb moodul Küttega alusplaadi juhtimine. Suurem enamik käske on viia printeri pea mingisse punkti. Selle jaoks on vaja kõigepealt teha printeri pea liikumisele profiil. Selle jaoks saadetakse printeri pea uue koordinaadi asukoht moodulisse Liikumisprofiili arvutus. See arvutab välja mis suunas ja millal peab samm-mootoriga sammu tegema. Kõik vajalik info sammu tegemiseks pannakse lingitud nimistusse ja seda infot hakkab ära kasutama moodul Impulsi generaator, mida kutsub moodul Main välja iga kindla aja tagant. Impulsi generaatori ülesandeks on ajastada samme õigesti. See saadab signaale moodulile Sammu järjekorra generaator, mis hakkab samm-mootorite mähiseid õiges järjekorras pingestama.

Printima hakkamiseks on vaja viia printeri peas olev küttekeha vajalikule temperatuurile. Moodul Main annab käsu moodulile Printeri pea juhtimine kas kütteha sisse või välja lülitada. Printeri pea juhtimine annab omakorda infot moodulile Main küttekeha temperatuuri kohta. Printeri pea samm-mootori juhtimist ajastab moodul Impulsi generaator, mis saadab samm-mootori töötamissagedusel impulsse moodulile Printeri pea juhtimine, mis omakorda saadab need edasi printeri pea juhtelektroonikale.

Enne printima hakkamist peab haarad viima algasendisse. Vastava käsu algasendisse viia annab moodul Main moodulile Liikumisprofiili arvutus. Samm-mootoreid seejärel liigutatakse, kuni moodul Piirlülitid saab signaali, et haar on liikunud algasendisse, mille peale moodul Main annab käsu liikumine lõpetada. Kui piirlülitid saadavad signaali teistel juhtudel, siis kutsutakse välja programmis katkestus ja vastav veateade saadetakse moodulisse Veatöötlus, mis annab käskluse mootorid seisma panna.

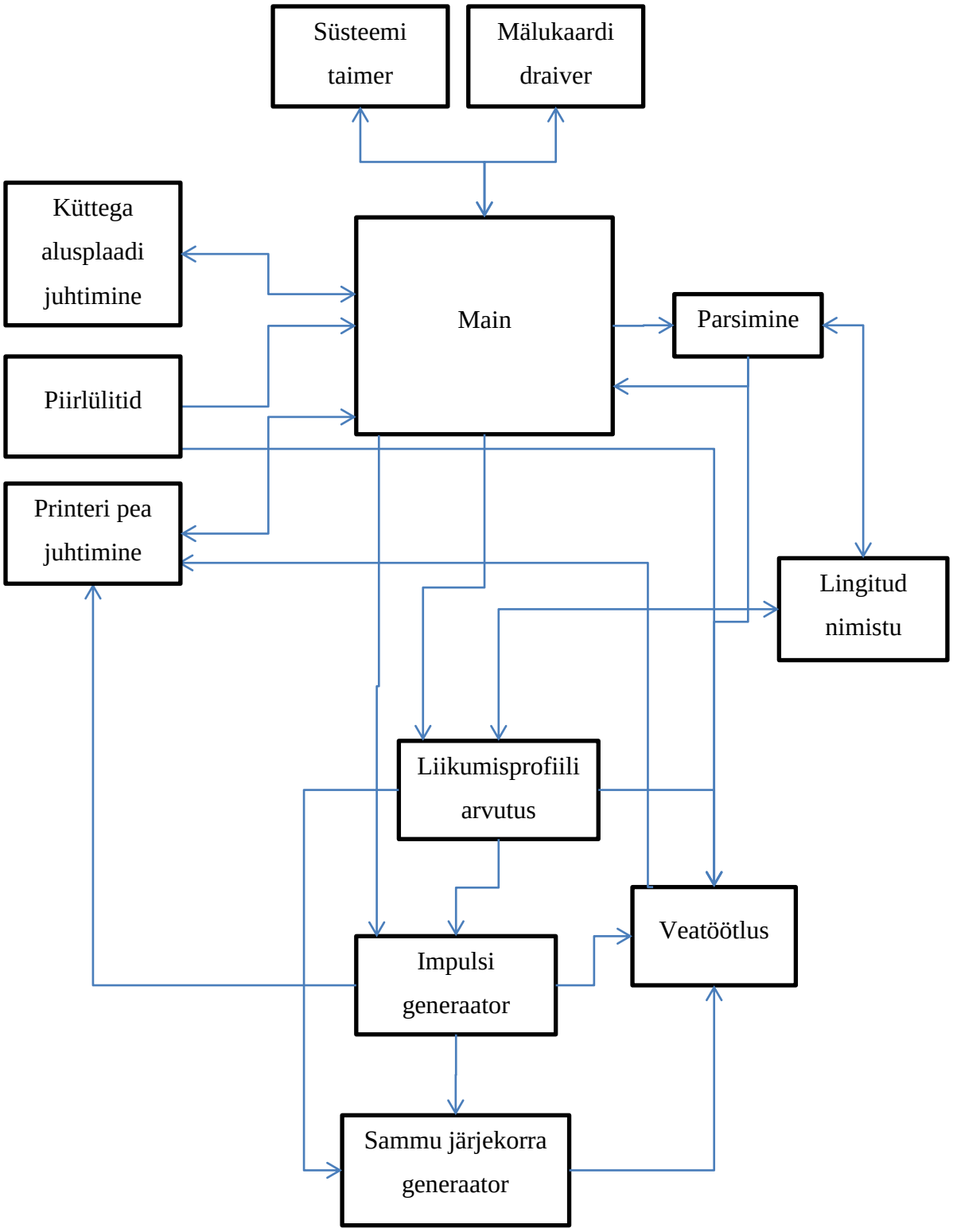
Moodul Main paneb enne käskude täitma hakkamist käima süsteemi taimeri, mis hakkab funktsioone välja kutsuma kindla aja tagant kasutades programmis katkestust. Funktsioonideks on näiteks küttega alusplaadi temperatuuri kontrollimine ja reguleerimine või uue sammu liikumisprofiili arvutus.

Suurem osa moodulitest kontrollib oma tööd ja kui tekib kusagil mõni viga, siis vastav veakood saadetakse moodulile Veatöötlus. Kuna mõne suure eseme printimine võtab pikka aega, siis see moodul võiks olla piisavalt tark, et veaga ise toime tulla ja printimist jätkata. Seda, mida moodul Veatöötlus vastava veasignaali juures teeb, peaks otsustama siis, kui süsteemi on katsetatud ja programmi kõige olulisemad osad, millega printerit juhtida, valmis kirjutatud.

Lõplik lahendus on jaotatud mooduliteks nii, et neid mooduleid võib põhjalikumalt või lihtsamalt realiseerida. See tähendab, et näiteks liikumisprofiili arvutuse võib esialgu konstantse kiirusega teha või mikrosamm-talitluse asemel kasutada poolsamm-talitlust. Lõplik programm võiks suuta teostada kõiki kolme talitlust ja muutuva kiirusega liikumist. Sellisel juhul saab seda paremini kasutada ka teiste projektide ja samm-mootoritega. Samuti võiks see võimalikult täpselt vastata NIST standardile [10].

Kasutajal peab olema võimalus juhtmooduli tööd mingil määral kontrollida. Näiteks nupule vajutades hakkab printer mälukaardilt infot lugema. Antud töös ei ole analüüsitud, mille kaudu kasutajaga suhtlus peaks toimuma, sest ei ole kindel, mida kasutaja peaks kontrollida saama. Osa käsklusi võivad arvatavasti olla seotud veatöötlusega. Esialgu võib kasutada mikrokontrolleri nuppe.

Lõplik lahendus vastab kõikidele lähtemäärangus olevatele punktidele.



Sele 4.1 Lõpliku lahenduse diagramm

5. PRAKTILINE OSA

Töö praktilise osana on osa koodist valmis kirjutatud programmeerimiskeeles C ja katsetatud Olimexi mikrokontrolleriga LPC-P2148. Arvuti on mikrokontrolleriga suhtlema pandud kasutades Seggeri programmaatorit J-Link seerianumbriga 268003858. Koodi silumiseks on kasutatud gdb serverit, mille tekitas programm SEGGER J-Link GDB Server V4.80c. Programmeerimise keskkonnana on kasutatud programmi Eclipse Standard/SDK Kepler Service Release 1 versiooni. Programmi kompileerimiseks on kasutatud GNU ja YAGARTO töövahendeid. Silumiseks vajaliku keskkonna ülesseadmiseks on jälgitud õpetusi, mis on saadaval internetilehelt viites [8].

5.1 Mälukaardi draiver

Mälukaardi draiveri realiseerimiseks on internetist alla laetud project, mis on tehtud LPC-2148 mikrokontrollerile ja mis kasutab ChaN-i FAT failisüsteemi[1]. See projekt on saadaval vabavarana viites [7].

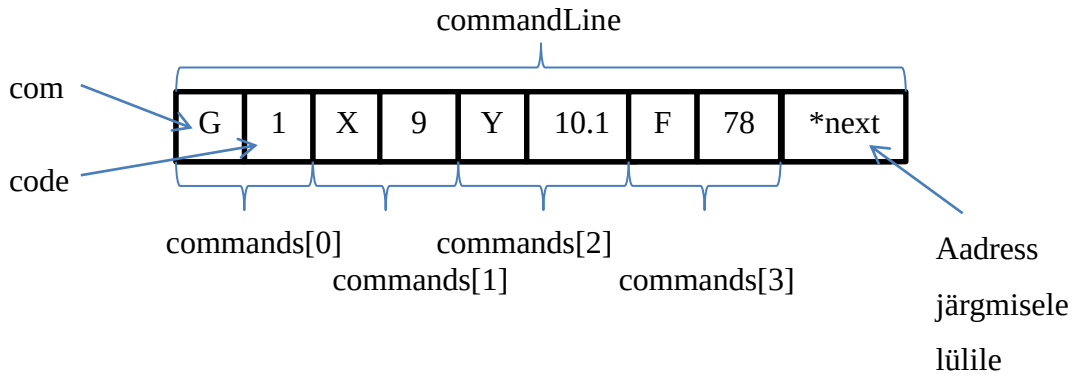
Mälukaardilt lugemiseks koodis peab faili süsteemi enne registreerima FatFs mooduli jaoks kasutades funktsiooni „f_mount“ [3]. Seejärel peab avama faili kasutades funktsiooni „f_open“ [4]. „f_read“ loeb mälukaardilt ette antud baitide arvu ja paneb selle *char* tüüpi massiivi. Seda massiivis olevat täherida peab parsima ja seejärel uuesti lugema järgmise hulga tähti.

5.2 Parsimine

Mälukaardilt loetud täherea parsimisega tegeleb moodul *Parsimine*. See moodul parsib mälukaardilt loetud koodi lingitud nimistusse, mille lüliks on struktuur nimega *commandLine*. Lingitud nimistu loomiseks on moodul nimega Lingitud nimistu. Struktuuris *commandLine* on kaks muutujat. Üks on *command* tüüpi struktuuri rida nimega *commands* ja teine on viit tüüpi muutuja järgmisele lülile nimega *next*. Struktuuris *command* on omakorda kaks muutujat: *char* tüüpi muutuja *com* ja *double* tüüpi muutuja *code*. Moodul parsib iga mälukaardilt loetud koodi rea uude struktuuri *commandLine*. Reas võib olla mitu elementi,

[1] ChaN-i failisüsteemist on rohkem juttu peatükis 3.2.1.

mis koosneb tähest ja numbrist. Rea elemendid salvestatakse struktuuri *command*. Näiteks kui mälukaardilt loetud täherida on „G1 X9 Y10.1 F78“, siis see parsitakse nii, nagu kujutab sele 5.1.



Sele 5.1. Parsitud koodi lüli

Samuti kontrollitakse, kas sisend tekstirida on RS274 standardile vastav kood. Funktsioonid kontrollimiseks on moodulis Veatöötlus.

5.3 Impulsi generaator

Impulsi generaator on kirjutatud nii, nagu on kirjas peatükis 3.1.5. Selles moodulis on funktsioon, mida moodul Main peab iga kindla aja tagant välja kutsuma.

5.4 Veatöötlus

Selles moodulis on kirjutatud funktsioonid, mis kontrollivad, kas sisend G-kood on vigadeta ja kas kõik käsud on antud süsteemis realiseeritud. Neid funktsioone kutsutakse välja parsimise käigus.

5.5 Kirjutatud funktsioonid

Tabel 5.1. Funktsioon f_readCommand

Deklaratsioon	void f_readCommand (char*);
Moodul	Parsimine

Sisend	Viit mälukaardilt loetud täherea mälupesale.
Väljund	-
Funktsioon	Eemaldada kommenteeritud osa sisendist ja jagada sisend ridadeks. Iga rida saata edasi funktsiooni f_parseLine.

Tabel 5.2. Funktsioon f_parseLine

Deklaratsioon	void f_parseLine(char*);
Moodul	Parsimine
Sisend	Viit mälukaardilt loetud ühe rea mälupesale.
Väljund	-
Funktsioon	Jagada rida käsklusteks, panna käsud ühte massiivi ja seejärel kutsuda välja funktsioon f_addCommandLine, mis paneb antud rea lingitud nimistusse.

Tabel 5.3. Funktsioon f_parseSingleCommand

Deklaratsioon	struct command f_parseSingleCommand(char);
Moodul	Parsimine
Sisend	Viit üksikule käskluse tähele.
Väljund	Struktuur command.
Funktsioon	Parsida sisend struktuuri command.

Tabel 5.4. Funktsioon f_addCommandLine

Deklaratsioon	void f_addCommandLine (struct command[], int);
Moodul	Lingitud nimistu
Sisend	Rea käsud ja käskude arv.
Väljund	-
Funktsioon	Tekitada uus lingitud nimistu lüli ja panna sisend informatsioon sinna lülisse.

Tabel 5.5. Funktsioon f_getHeaderCommandLine

Deklaratsioon	struct commandLine* f_getHeaderCommandLine(void);

Moodul	Lingitud nimistu
Sisend	-
Väljund	Parsitud G-koodi lingitud nimistu esimese lüli aadress.
Funktsioon	Tagastada parsitud G-koodi lingitud nimistu esimese lüli aadress.

Tabel 5.6. Funktsioon f_deleteHeader

Deklaratsioon	void f_deleteHeader(void);
Moodul	Lingitud nimistu
Sisend	-
Väljund	-
Funktsioon	Kustutada parsitud G-koodi lingitud nimistu esimene lüli.

Tabel 5.7. Funktsioon f_addMotorMovement

Deklaratsioon	void f_addMotorMovement (enum eMotors, uint32_t,enum eDirection);
Moodul	Lingitud nimistu
Sisend	Mootori number, viivituste arv ja liikumise suund.
Väljund	-
Funktsioon	Panna sisend informatsioon antud mootori kohta olevasse lingitud nimistusse.

Tabel 5.8. Funktsioon f_getMotorMovement

Deklaratsioon	struct motorMovement* f_getMotorMovement(enum eMotors);
Moodul	Lingitud nimistu
Sisend	Mootori number.
Väljund	Viit antud mootori liikumist iseloomustavale lingitud nimistu esimesele lülile.
Funktsioon	Tagastada viit antud mootori liikumist iseloomustavale lingitud nimistu esimesele lülile.

Tabel 5.9. Funktsioon f_checkCommand

Deklaratsioon	int f_checkCommand(char);

Moodul	Veatöötlus
Sisend	Täht, mis iseloomustab G-koodi käsku.
Väljund	1, kui toetab antud käsklust vastasel korral kutsutakse välja funktsioon f_errorHandling.
Funktsioon	Kontrollida, kas programm toetab antud käsklust.

Tabel 5.10. Funktsioon f_checkNumber

Deklaratsioon	int f_checkNumber(char);
Moodul	Veatöötlus
Sisend	Täht, mis iseloomustab numbrit.
Väljund	1, kui on number. 2, kui on punkt. 0, kui ei ole kumbki eelnevatest.
Funktsioon	Kontrollida, kas antud täht on number.

Tabel 5.11. Funktsioon f_checkStringNumber

Deklaratsioon	void f_checkStringNumber(char*);
Moodul	Veatöötlus
Sisend	Viit tähereale
Väljund	-
Funktsioon	Kontrollida, kas antud täherida on number.

Tabel 5.12. Funktsioon f_ performSteps

Deklaratsioon	void f_performSteps(void);
Moodul	Impulsi generaator
Sisend	-
Väljund	-
Funktsioon	Ajastada samme sammu järjekorra generaatori jaoks.

KOKKUVÕTE

Töö eesmärgiks on uurida ja leida lahendus Mehhatroonikainstituudi laboris olevale 3D printeri juhtmooduli programmile. Selleks on kõigepealt peatükis kaks uuritud põhjalikumalt antud printerit ja võimalikke probleeme, mis võivad tekkida ning mida peab töös käsitlema. Peatüki kaks põhjal on koostatud programmi lähtemäärang.

Juhtmooduli kõige olulisemaks osaks on telgede liigutamine ja seetõttu on peatükis kolm esimesena uuritud, kuidas samm-mootoreid juhtida. Alustatud on nende juhtimise lihtsustatud põhimõttest. Edasi on uuritud mootori poolt tekitatavat momenti ja printeri positsioneerimise staatilist viga. Üks võimalus staatilist viga vähendada on kasutada teistsugust juhtimise talitlust ja seetõttu on neid analüüsitud järgmisena. Võrreldud on talitluste positsioneerimise täpsust, energiakulu, tekitatud pöördemomenti, hindu ning keerukust. Seda informatsiooni saab ära kasutada, kui hakatakse tegema katseid 3D printeriga ja selgub, et näiteks samm-mootorid ei suuda piisavalt pöördemomenti tekitada lihtsama talitlusega või samm-mootorid liiguvad liiga robustselt. Seejärel on uuritud samm-mootorite kiiruse piiranguid staatilises olekus. Seda saab kõige paremini teada piirmomendi/kiiruse graafikult ja töös on antud juhised, kuidas seda graafikut leida.

Antud projektis kasutataval 3D printeril puudub tagasiside samm-mootoritelt ja seega on uuritud, kuidas neid juhtida avatud ahelaga. Töös on näidatud, kuidas teostada konstantse kiirusega liikumist. Selle põhjal saab 3D printeri esialgu konstantse kiirusega tööle panna. Samm-mootori väga oluliseks näitajaks on käivitussagedus, mis näitab, millise kiirusega võib mootori nullist liikuma panna. Antud töös on kirjeldatud, kuidas seda sagedust ligikaudselt arvutuslikult leida. Selle sageduse kaudu saab teada, kas programm peab välja arvutama kiirendus- ja aeglustusprofiili. Samuti on vaja käivitussagedust nende profiilide arvutamiseks.

Samm-mootorite juhtimise kokkuvõttena on kirjutatud, kuidas rakendada avatud ahelat 3D printeris. Selles peatükis on samm-mootorite juhtimine jaotatud moodulitesse, on uuritud nende sisendeid, väljundeid ja funktsiooni ning millist informatsiooni need omavahel vahetavad.

Pärast samm-mootorite juhtimist on analüüsitud erinevaid G-koodi juhtmoodulisse edastamise võimalusi. Selle põhjal on otsustatud kasutada mälukaarti ja seejärel kirjeldatud, kuidas mälukaardi draiverit realiseerida. Selle kirjelduse põhjal on töö praktilises osas mälukaardi draiver mikrokontrolleris tööle saadud. Seejärel on kirjutatud G-koodi parsimisest ja pakutud võimalik lahendus parsimise väljundile. Selle põhjal on kirjutatud funktsioonid, mis G-koodi lingitud nimistusse parsivad.

Kolmanda peatüki lõpus on kirjas küttega alusplaadist, piirlülititest ja printeri peast. Need kolm komponenti on jaotatud eraldi moodulitesse ja töös on kirjas, mida need moodulid teevad, millist informatsiooni vahetavad ja kuidas neid rakendada.

Peatükis neli on toodud välja kokkuvõttev programmi lahendus. Selles on näidatud, kuidas on lõplik programm moodulitesse jaotatud. Samuti on seal kirjeldatud kogu programmi tööd. Selle peatüki põhjal saab kirjutada mooduli, mis hakkab kõikide teiste moodulite tööd juhtima.

Süsteemi ohutuse tagamiseks ja vigade töötlemiseks on loodud eraldi moodul nimega Veatöötlus. Terve töö käigus on kirjeldatud suhtlust selle mooduliga, aga ei ole täpsustatud, kuidas käituda erinevate vigade korral. Korralikuma vigade analüüsi peaks tegema siis, kui on juhtmooduli elektroonika osa samuti analüüsitud.

Viiendas peatükis on kirjas, kui kaugele on programmi kirjutamisega hetkel jõutud. Seal on kirjutatud, kuidas kasutada mälukaardi draiverit ja kuidas toimub G-koodi parsimine. Samuti on seal välja toodud funktsioonid, mis on valmis kirjutatud.

Järgmisena peaks analüüsima põhjalikult juhtmooduli elektroonika osa ja tegema otsuse, mida peaks koodina ja mida riistvaraliselt realiseerima. Samuti tuleb uurida, kuidas kasutaja saaks suhelda juhtmooduliga. Seejärel tuleks 3D printerit katsetama hakkama ja selle käigus välja selgitama, millist samm-mootorite talitlust ja liikumisprofiili kasutada.

Antud töö vastab sissejuhatuses formuleeritud ülesandele. Sellega on leitud programmiline lahendus 3D printeri juhtimiseks ja selgitatud samm-mootorite tööd. Samuti on selgitatud, kuidas printerit kõigepealt lihtsamalt ja seejärel optimaalsemalt tööle saada.

SUMMARY

The purpose of this thesis is to research and find a solution for a microcontroller program which will control a 3D printer that is in the Mechatronic institute. The printer is examined closely in the second chapter. Furthermore some problems that may arise with a large 3D printer are brought out. Delimitation is made according to the analysis in chapter two.

The main purpose of the microcontroller is to move the printer axes and therefore controlling stepper motors is explained next in chapter three. At first the simple principle of controlling the motors is shown. The next thing that is discussed is how they produce torque and what their static error is. One way to reduce static error is to change the excitation scheme and therefore different schemes are discussed next. Positioning accuracy, energy consumption, torque, price and complexity of different schemes are compared. That information can be used when during the testing of the printer the motors do not produce enough torque or their movement is too coarse. The maximum speed of the stepper-motors is investigated next. That speed can be found from the torque/speed graph of the motors and instructions are given on how to make that graph.

The printer that is used in this project does not have feedback from the motors and therefore only open-loop control is examined. In this thesis it is shown how to run the motors at constant speed. This information can be used to initially control the extruder at constant speed, because that is much simpler than running the motors at variable speed. A very important attribute of a stepper motor is its pull-out torque, which shows the stepping the motors can be started without losing any steps. It is shown in this thesis how to approximately calculate the pull-out torque, which indicates whether the program needs to calculate velocity profiles for the motors. Furthermore the pull-out torque is needed to calculate these profiles.

How to implement the open-loop control is written to summarize the stepping motors part. The controlling of the motors is separated into modules. In addition the inputs, outputs, function and what information the modules exchange is written next.

How to get the G-code to the microcontroller is analysed next. The conclusion of the analysis is that a memory card is best suited for that. In addition how to implement a memory card

driver is described. In the practical part of the work the memory card driver is implemented and working in the microcontroller based on that description. It is then written how to parse the G-code and a possible solution for the output of the parsing is given. Functions that parse a G-code to a linked list are written based on that solution.

It is written about the heated plate, limit switches and extruder at the end of the third chapter. Those three components are divided into separate modules and what those modules do, what information do they exchange and how to implement them is discussed.

A final solution for the program is presented in chapter four. In that it is shown how the program is modulated. Furthermore how the program runs is described. Chapter four can be used to create a module that will control the work of all other modules.

For safety and error handling a separate module is added to the program. How other modules communicate with the error handling module is discussed all through the thesis, but it is not specified what the error handling module does in each specific case. A more thorough error analysis needs to be done when the controller electronic part is also analysed.

The fifth chapter explains how much of the program is written. It is written in that chapter how to use the memory card driver and how the program parses the G-code. Furthermore all the functions and their descriptions are written in that chapter.

The electronic part of the controller should be analysed next and what to implement with a code and what to implement electronically should be decided. Furthermore how the user communicates with the controller should be analysed. Then the printer should be tested and based on that decide what sort of excitation scheme and velocity profile should be used.

This thesis covers all the points written in the introduction. A programmable solution for a 3D printer is discussed and presented. Furthermore how to control a stepping motor is investigated. In addition it is discussed how to implement the solutions in an easier or more thorough manner.

KASUTATUD KIRJANDUS

[1] A3979 microstepping motor driver. [WWW] http://www.farnell.com/datasheets/1756796.pdf (9.05.2014)

[2] Acarnley, P. Stepping Motors : a guide to theory and practice. 4th edition. London, United Kingdom : The Institution of Electrical Engineers, 2002.

[3] f_mount. [WWW] http://elm-chan.org/fsw/ff/en/mount.html (11.04.2014)

[4] f_open. [WWW] http://elm-chan.org/fsw/ff/en/open.html (11.04.2014)

[5] FatFs – Generic FAT File System Module. [WWW] http://elm-chan.org/fsw/ff/00index_e.html (11.04.2014)

[6] Geelhoed, A. 3D Printing: The New Industrial Revolution That Will Change the Landscape of Credit Risk Management. [WWW] http://web.b.ebscohost.com/ehost/detail?sid=25e4ac2f-aa3d-41fc-bbfd-b1c8e3f99c15%40sessionmgr198&vid=1&hid=119&bdata=JnNpdGU9ZWhvc3QtbGl2ZSZzY29wZT1zaXRl#db=f5h&AN=94812365 (17.05.2014)

[7] Interfacing ARM controllers with Memory-Cards. [WWW] http://siwawi.bauing.uni-kl.de/avr_projects/arm_projects/arm_memcards/index.html (11.04.2014)

[8] J-Link GDB Server guide. [WWW] http://www.emb4fun.de/archive/gdbserver/index.html (15.03.2014)

[9] Jones, Douglas, W. Microstepping of Stepping Motors. – *Stepping Motors*, 1995. [WWW] http://homepage.cs.uiowa.edu/~jones/step/micro.html (26.04.2014)

[10] Kramer, T, R. Proctor, F, M. Messina, E. The NIST RS274NGC Interpreter - Version 3. Standard, 2000, version 3, 12. [WWW] http://www.nist.gov/customcf/get_pdf.cfm?pub_id=823374 (6.04.2014)

[11] Microstepping: Myths and Realities. [WWW] http://www.micromo.com/microstepping-myths-and-realities.aspx (3.05.2014)

[12] One eighth Microstep/Step Operation. [WWW] http://softsolder.files.wordpress.com/2011/05/allegro-a3977-microstepping-current-waveforms.png (3.05.2014)

[13] Parker, M. Holger, K. Projekti aruanne : õppeaine aruanne. Tallinn, Tallinna Tehnikaülikool, 2012.

[14] Rannamäe, M. Suuregabariidiline 3D printer : bakalaurusetöö. Tallinn, Tallinna Tehnikaülikool, 2012.

[15] Samm-mootoriga ajamid. [WWW] http://www.ene.ttu.ee/elektriajamid/oppeinfo/materjal/AAV0040/Samm-mootoriga_ajamid.pdf (27.04.2014)

[16] Singly linked lists in C. [WWW] http://www.cprogramming.com/tutorial/c/lesson15.html (11.04.2014)

[17] SPI and SD cards. [WWW] http://www.dejazzer.com/ee379/lecture_notes/lec12_sd_card.pdf (11.04.2014)

[18] Stepper Motors: Principles of Operation. [WWW] https://www.pc-control.co.uk/step-motor.htm (20.04.2014)

[19] Täiturid Töösusautomaatikas / E. Brindfeldt, E. Pettai, H. Hõimoja, V. Beldjajev. Esimene trükk. Tallinn : 2011.