Kaur Samuel Kannel

ROBOTISEERITUD TOOTMISLIINI POSITSIOONI ARENDAMINE

LÕPUTÖÖ

Tehnikainstituut

Robotitehnika õppekava

Juhendajad: Villu Lõhmus, Harti Vait

Tallinn 2024

AUTORI DEKLARATSIOON JA LIHTLITSENTS

Mina, Kaur Samuel Kannel

annan Tallinna Tehnikakõrgkoolile (edaspidi kõrgkool) tasuta loa (lihtlitsentsi) enda loodud teose

ROBOTISEERITUD TOOTMISLIINI POSITSIOONI ARENDAMINE

1) reprodutseerimiseks eesmärgiga seda säilitada ja teha üldsusele kättesaadavaks Tallinna Tehnikakõrgkooli digiarhiivi DSpace kaudu;

2) reprodutseerimiseks pärast piirangu lõppu juhul, kui instituudi direktori korraldusega on kehtestatud lõputöö avaldamisele tähtajaline piirang.

Olen teadlik, et nimetatud õigused jäävad alles ka autorile ja kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest tulenevaid ega muid õigusi.

Autorideklaratsioon

Mina, Kaur Samuel Kannel

tõendan, et lõputöö on minu kirjutatud. Töö koostamisel kasutatud teiste autorite, sh juhendaja ja iseenda varasematele teostele, on viidatud õiguspäraselt. Kõik isiklikud ja varalised autoriõigused käesoleva lõputöö osas kuuluvad autorile ainuisikuliselt ning need on kaitstud autoriõiguse seadusega.

(allkirjastatud digitaalselt)

Villu Lõhmus / Harti Vait

Töö vastab lõputööle esitatavatele nõuetele.

(allkirjastatud digitaalselt)

Lõputöö on kaitsmisele lubatud instituudi direktori korraldusega.

(allkirjastatud digitaalselt)

(kuupäevad digiallkirjades)

SISUKORD

AUTORI DEKLARATSIOON JA LIHTLITSENTS 2
LÜHENDID 4
SISSEJUHATUS 5
1. PROBLEEMI KIRJELDUS 6
1.1 Toodetav detail 8
1.2 Nõuded 9
1.3 Mehaanika kirjeldus 10
1.3.1 Robotkäe otsaefektorid 10
1.3.2 Rakis 11
2 KASUTATUD TEHNOLOOGIATE KIRJELDUS 14
2.1 Python 14
2.2 Aurora Vision Studio 15
2.2.1 Mallide sobitamine 15
2.2.2 Süvaõpe 17
3 TOOTMISTSÜKLI KIRJELDUS 19
4 KÄEPIDEME RASVATAMISE MOODUL 22
5 UPPER PLATE DETAILI ETTEANDUR 25
5.1 Probleemi kirjeldus 25
5.2 Asukoha, nurga ja orientatsiooni määramine 26
5.3 Etteanduri kontroller 29
5.4 L-versioon 30
6 OHZ DETAILI ETTEANDUR 32
6.1 Probleemi kirjeldus 33
6.2 Asukoha ja nurga määramine 34
6.3 Orientatsiooni määramine 37
6.4 Koordinaatteljestikkude konverteerimine 39
7 LINEAARTELG 41
KOKKUVÕTE 45
SUMMARY 46
VIIDATUD ALLIKAD 47

LÜHENDID

UR – *Universal Robots* (lahenduses kasutatud robotkäe nimi)

Cobot – *Collabirative robot* / koostöörobot

RPC – *Remote Procedure Call* / protseduuri kaugkäivitus

XML – Extensible Markup Language / jagatav märgistuskeel

AI – *Artificial Intelligence* / tehisintelekt (kattev termin kõikide keeruliste algoritmide kohta, selles töös peamiselt tähendab neurovõrku)

IO – Input, Output / sisend, väljend

IP – Internet Protocol / interneti protokoll

SISSEJUHATUS

Käesoleva lõputöö keskmes on tarkvara arendamine ja testimine tootmisliini ühele positsioonile. Antud tootmisliin koosneb viiest positsioonist, millest kolm on automatiseeritud. Autor on teinud tööd kõigi kolme automatiseeritud positsiooni juures, kuid lõputöö keskendub konkreetselt ühele neist, positsioonile nr 3. Tootmisliin ise on loodud Optimo Robotics OÜ poolt ning tellitud Plastone OÜ-st. Autor tegutses Optimo Robotics-is lõputöö koostamise perioodil.

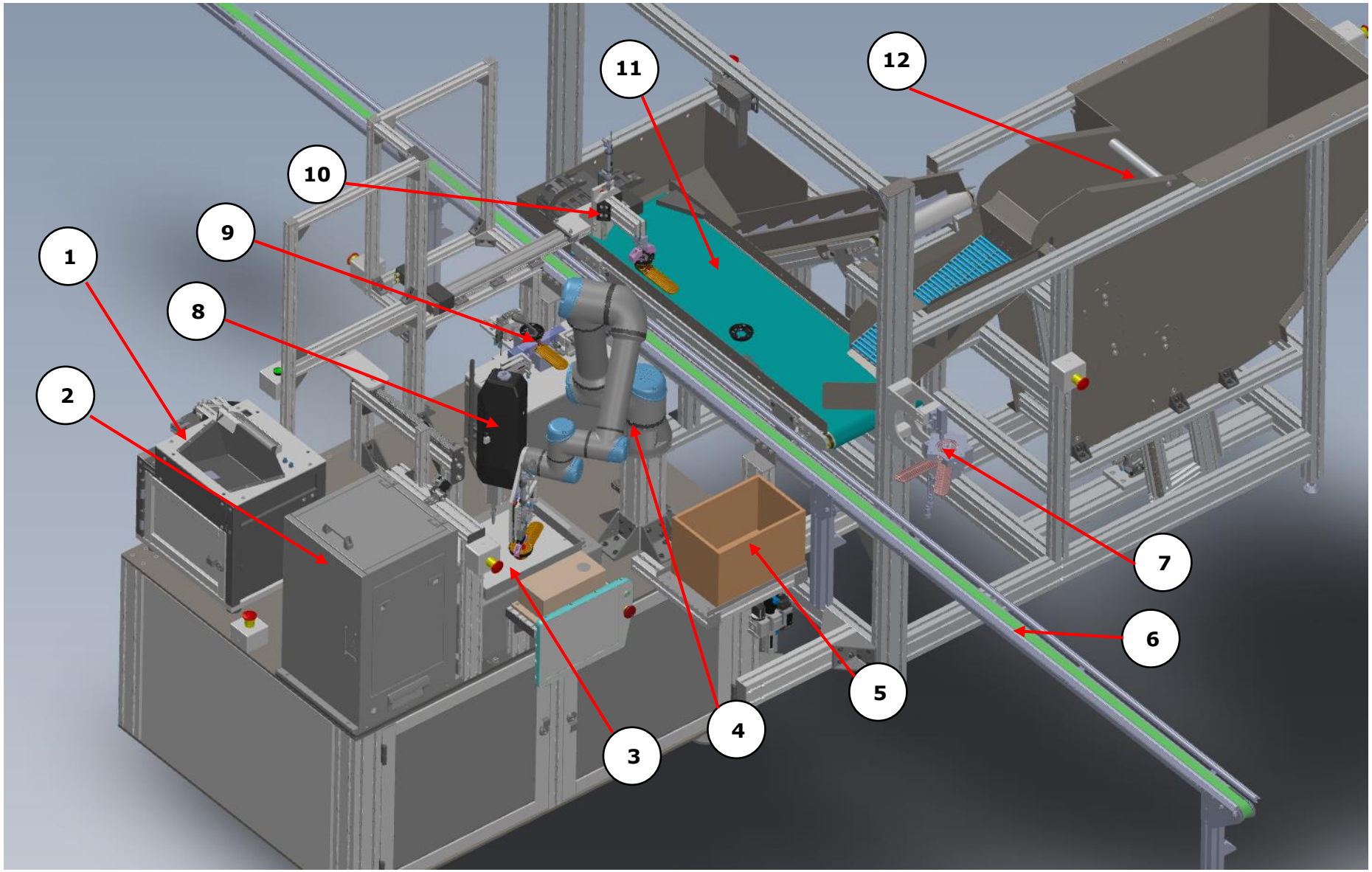
Lõputöö raames ei käsitleta tootmisliini mehaanikakomponentide projekteerimist ja valimist, kuna need olid ettevõttes juba varem välja töötatud mehaanikute poolt. Lõputöö fookus on suunatud tarkvaralisele aspektile, positsiooni 3 juhtimisele. Tootmisliin ise toodab käsilülitusseadet, mis on leidnud kasutust näiteks elektrikilpide pealülitina.

1. PROBLEEMI KIRJELDUS

Probleem, mida autor lõputöö raames lahendab, on tarkvara puudumine automaatikamoodulil. Automaatikamoodul on osa tervest tootmisliinist, järjekorras kolmas positsioon. Positsioon tegeleb toodetava detaili liikuvate osade kokkupanemisega ja pärast nende kokku kruvimisega. Kõik detailid, mida positsioon kasutab toote kokkupanemisel, on mõeldud positsiooni poolt võtmiseks, kasutades nendeks ettenähtud automaat-etteandureid. Automaatetteandurite tarkvara arendus on autori tööülesanne ja nende tarkvara puudumine on osa lõputöö probleemist. Eelnev positsioon tootmisliinil on manuaalne ehk see, mis saab töötada ainult operaatoriga ning eelneva positsiooni lõpptoode on ka 3. positsiooni üks koostedetailidest.

Joonis 1. On näha lõputöös käsitletud automaatikapositsiooni illustratsiooni, olulisemad nummerdatud komponendid on järgnevad:

1. kruvifeeder ehk kruvide etteandur;
2. ohz detaili etteandur;
3. paindlaud, ohz detaili etteandur;
4. UR robotkäsi;
5. praaktoodete kast;
6. tootmisliini peakonveier;
7. rasvatamise moodul;
8. automaatkruvikeeraja;
9. rakis;
10. lineaartelg;
11. upper plate detaili etteandur;
12. upper plate detaili kolu.

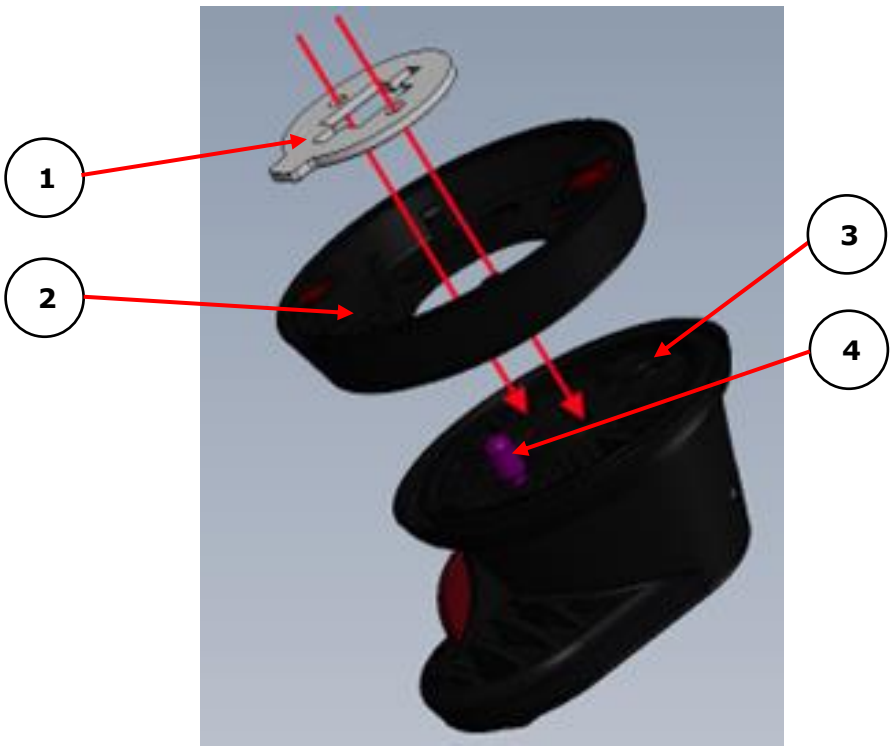


Joonis 1. Automaatikapositsioon nr. 3

1.1 Toodetav detail

Toode, mida terve tootmisliin toodab, on käepide, mida saab kinnitada lülitusseadmete otsa. Kasutusaladeks on näiteks elektrikilpide pealüliti või releede ja kontaktide manuaalne lülitus. Automaatikamoodul on ainult üks etapp terves detaili tootmisliinis ja keskendub ainult spetsiifilisele ülesandele - kolme detaili kokku kruvimisele (Joonis 2). Nendeks detailideks on:

- Käepide (nr 3), mis on eelneva tootmisliini positsiooni lõpptoode ja mille sees on operaatori poolt asetatud vedruga tihvt. Mainitud tihvt (nr 4) ei tohi protsessi ajal käepideme seest välja kukkuda. Käepidemes on sooned, kuhu sisse sisestatakse järgmine detail. Mainitud sooned peavad olema ära rasvatatud, et tagada toote korrektne töötamine.
- *Upperplate* (nr 2), toote keerleva osa detaili on kahes erinevas variandis ja moodul peab suutma mõlemat töödelda, L-versiooni ja J-versiooni. L-versioonil on operaatorite poolt sisestatud lisaks üks manuaalselt rasvatatud tihend, sellega koostatud lõpptoodet toodetakse harvem ja puudub enda automaatne etteandur.
- OHZ (nr 1), metallist seib, millel on väike väljaulatuv lest. Eesmärgiks on lubada käepidemel keerata ainult teatud mahus. Seda detaili on kolmes erinevas variatsioonis: kahe kruviauguga sirge lestaga/kõvera lestaga ja nelja kruviauguga versioon, kõik variatsioonid tulevad samast automaatsest etteandurist.



Joonis 2. Detaili koostamisjuhend automaatikamoodulile

Pärast eelmainitud detailide kokkupanekut peab automaatikapositsioon need kokku kruvima. Kruvimisel peab jälgima, et kruvimoment ei varieeruks lubatavast tasemest rohkem kui 10%, mis oli käesoleval tööl 1Nm.

1.2 Nõuded

Automaatikamooduli peaülesandega kaasnevad ka nõuded, mis olid kliendi poolt ette antud ja mida automaatikapositsioon peab koostamise ajal jälgima või ei tohi ületada.

- müratugevus alla 85dB**;**

Tootmisliin on poolautomaatne ehk positsioonides 2 ja 4 tegutsevad operaatorid ning tehases, kuhu tootmisliin paigaldatakse, ei ole nõuet töötajatel kasutada mürakaitse varustust. Sellepärast kehtib nõue, et terve tootmisliini maksimaalne müratugevus peab jääma alla 85dB, et mitte kahjustada operaatorite või läheduses viibivate inimeste kuulmist.

- tootmiskiirus 3 toodet minutis;

Tellija nõue on, et liin peab tootma minimaalselt 3 toodet minutis. Selle nõude täitmiseks ei olnud võimalik seadistada robotkätt cobot režiimi, kuna siis oleks robotkäe liikumiskiirus liiga aeglane. Tootmiskiiruse nõuet oli isegi ilma cobot režiimita väga raske täita ja pidi kasutama kõiki võimalusi säästmaks võimalikke millisekundeid. Peamiseks pudelikaelaks jäid etteandurid, mis võisid vahepeal sobiva detaili ettevalmistamiseks võtta palju kauem aega, kui üks 20 sekundiline tsükkel. Kuid läbi kavalate puhvervaru süsteemide ning võimalikult palju paralleelseid protsesse kasutades, sai see nõue ka täidetud mõningate eranditega.

- tootmise efektiivsus 98%;

Tootmise efektiivsus all on mõeldud nõuet, et moodul ei toodaks praaki. Siia alla ei kuulu toodetud praak, mis on väliste põhjuste pärast loetud praagiks, nt. defektiga etteandurisse sattunud koostamisdetailid või operaatorite vead. Tootmise efektiivsus 98% käib kõigi kolme automaatikapositsiooni kohta tootmisliinil kokku, mis tähendab, et tegelik lubatud efektiivsus ühel positsioonil on palju kõrgem.

- turvalisus.

Turvalisus on kõige tähtsam nõue. Tegu ei ole koostöörežiimis oleva automaatika positsiooniga, mis tähendab, et roboti kiirus osades kohtades ületab isegi 1 meetrit

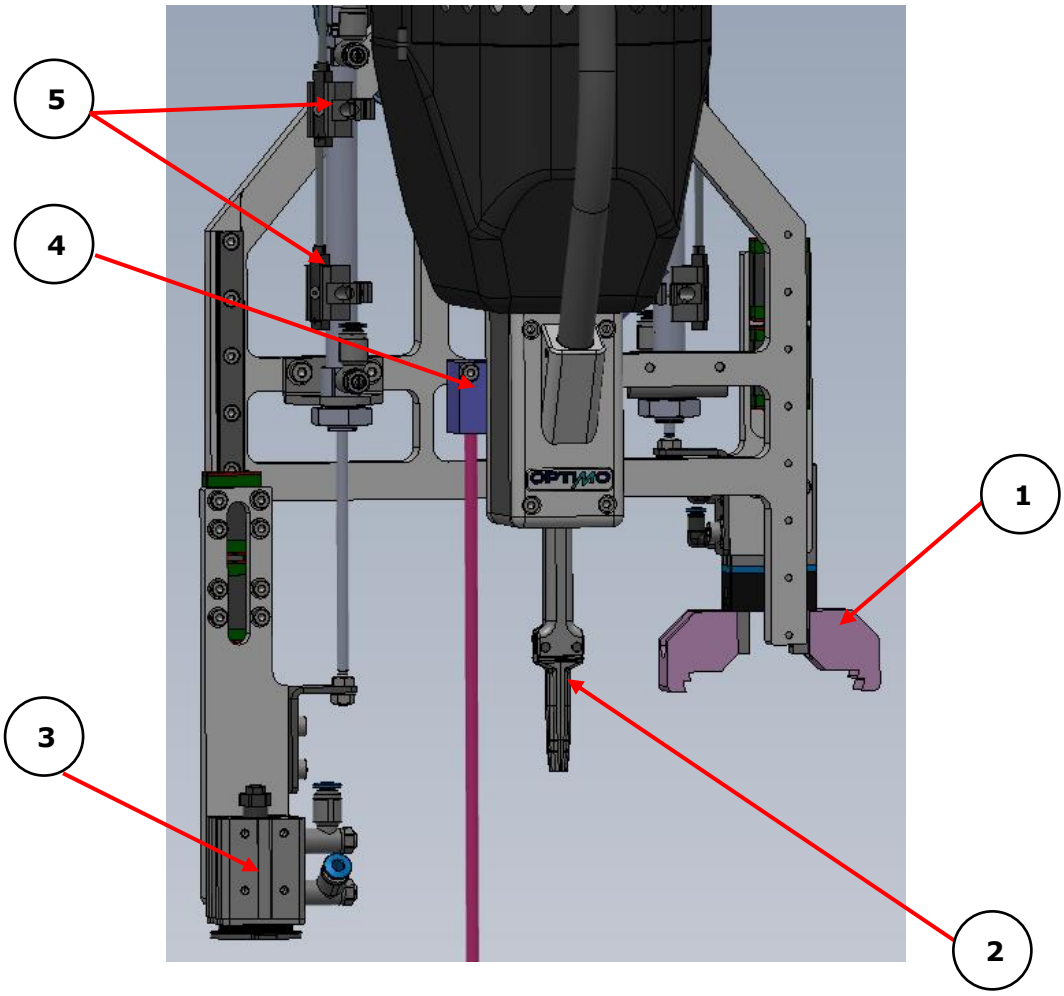
sekundis. Kui inimese jäsemed või muud obstruktsioonid peaksid sattuma roboti liikumise ette, võib kahju neile olla väga suur. Terve automaatikapositsioon oli just selle nõude tõttu ümbritsetud kas turvaaiaga või turvakardinatega. Nõude tegi raskeks see, et positsioon on poolautomaatne, operaator peab käega panema roboti tööalasse ühe detaili.

1.3 Mehaanika kirjeldus

Mooduli mehaaniline projekteerimine, ehitus ja komponentide valik ei kuulu lõputöö skoopi. Autor alustas projektiga siis, kui see oli juba projekteerimisfaasist väljas. Vajalik oli arendada tarkvara lahendus, mis töötaks juba varasemalt välja valitud komponentidega. Komponendid, mis ei kuulu ühegi etteanduri alla, on kirjeldatud järgnevates peatükkides.

1.3.1 Robotkäe otsaefektorid

Tootmistsükli koostusülesandeid täidab UR robot, millele oli mehaanikute poolt disainitud selleks vajalikud robotkäe otsaefektorid. (Joonis 3)



Joonis 3. UR robotikäe otsaefektorid

1. käepideme ja upperplate haarats;

Haaratsi kuju on disainitud erineva läbimõõduga hammastega, mis võimaldab haarata nii käepidet kui ka upperplate-i sama otsaefektoriga. Silindriks on kasutatud Festo paralleelset haaratsit, tootekood DHPS-10-A. Haaratsi kohale on kinnitatud ka optiline sensor, mis on võimeline nägema, kas haaratsil on midagi käes või mitte. Tänu sellele on võimalik sooritada lisakontroll, kas haarats on korrektselt haaranud detaili.

2. Optimo S2 Mini Robotic Screwdriver;

Optimo Robotics poolt disainitud automaatkruvikeeraja, tegemist on selleks projektiks arendatud uue kruvikeeraja versiooniga, mis on odavam kui S1 kruvikeeraja, aga kasutab Z-telje liikumiseks silindrit, mitte kuulkruvi.

3. OHZ detaili magnetgripper;

OHZ detaili haaratsiks sai mehaanikute poolt valitud magnetgripper. Kõik OHZ versioonid on valmistatud ferromagnetilistest metallist, mis tähendab, et neid on võimalik haarata kasutades staatilisi magneteid. Magnetgripper liigutab staatilist magnetit üles-alla, kasutades pneumaatilist kolbi.

4. lasersensor;

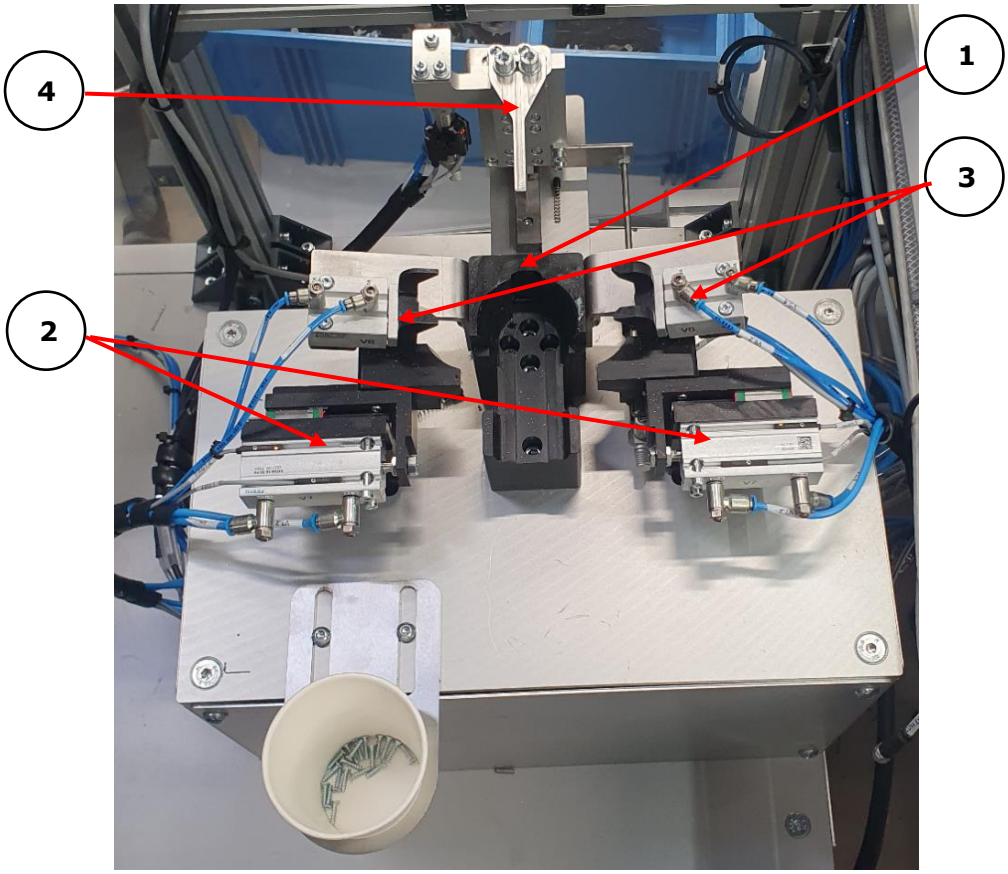
Lasersensor sai muretsetud algselt ainult kruvipeade kõrguste kontrollimiseks, kuid leidis kasutust paljudes muudes kvaliteedikontrolli operatsioonides. Näiteks on võimalik kontrollida, kas rakis, kus toimub komplekteerimine, on tühi, enne kui sinna asetada uus käepide või kas upperplate on korrektselt käepideme soontesse asetatud.

5. halli efekti andurid.

Igale kolvile, millele on haaratsid kinnitatud, on asetatud ka kaks halli efekti sensorit mõlemasse silindri piirpositsiooni. Sensorid võimaldavad robotil aru saada, millal on kolvid täielikult pikendatud või tagasi tõmmatud.

1.3.2 Rakis

Rakis on koht, kus robotkäsi komplekteerib lõpptoodet ja kus kruvitakse toote liikuvad detailid omavahel kokku. Rakis on eelnevalt projekteeritud mehaanikute poolt. Rakise peamine ülesanne on hoida komplekteeritavat toodet turvaliselt paigal, et tootetav detail ei liiguks ajal, mil robot töötab. Joonis 4 on näha rakisele kinnitatud erinevaid kolbe, mis abistavad robotit toote komplekteerimisel:



Joonis 4. Automaatikapositsiooni rakis

1. käepideme pesa;

Pesa, kuhu robot asetab käepideme ja komplekteerib kõik teised detailid selle peale. Pesa kuju võimaldab seda pesa kasutada erinevate pikkustega käepidemete komplekteerimisel. Samuti võimaldab pesa disain lineaartelje abil komplekteeritud käepideme välja tõsta, kui tootmine on lõpetatud.

2. käepideme hoidjad;

Hoidjad võimaldavad hoida käepidet kinni, likvideerides riski, et see hakkab tootmise ajal liikuma. Teiste detailide asetus ja hiljem nende kokku kruvimine on suurt täpsust nõudev töö, kus iga väike käepideme asukoha nihkumine võib lõppeda praagi tootmisega.

3. upperplate hoidjad;

Algselt ei olnud projekteeritud rakisesse upperplate-i hoidjat, kuid positsiooni programmeerimise käigus leidis autor, et juhul kui upperplate-i ei hoia midagi kinni, siis ebaõnnestub kruvimine väga tihti. Upperplate-i hoidja sai lisatud rakise külge siis, kui projekt oli juba programmeerimise faasis.

4. OHZ hoidja või „giljotiin".

OHZ hoidja kujutab endast terava servaga metalist kirvest (sellest tulev hüüdnimi giljotiin, kuidas autor nimetas detaili programmikoodis). Hoidja surub OHZ-i täpselt kurviaukude kohale kinni ja ei lase detailil liikuda kruvimise ajal.

2 KASUTATUD TEHNOLOOGIATE KIRJELDUS

Paljud tehnoloogiad ja tööriistad, mida autor kasutas lõputöö raames, ei ole autori poolt arendatud. Neid tööriistu, algoritme ja tehnoloogiaid kasutades arendas autor oma lahenduse, et säästa aega. Kõige tihedamalt kasutatud tööriistade ja tehnoloogiate kirjeldused on toodud välja järgmises peatükkides.

2.1 Python

Phyton on kõrgtaseme programmeerimiskeel, mida tuntakse selle lihtsuse, loetavuse ja mitmekülgsuse poolest. Selle süntaks rõhutab loetavust ja võimaldab väljendada käskusid võimalikult vähese koodiridade abil, muutes selle heaks valikuks nii algajatele kui ka kogenud arendajatele. Phyton on avatud lähtekoodiga mis soodustab kogukonda, mis pidevalt panustab selle kasvu ja arengusse. Projektis kasutas autor Phytonit, kus võimalik, kuna oli sellega varasemalt palju tööd teinud ja ettevõttes on see standard. [1]

Pythonil on ulatuslik standardteek ja lai valik kolmandate osapoolte pakette, mis katavad laia valiku rakendusalasid, sealhulgas andmeanalüüs, masinõpe, tehisintellekt, automatiseerimine ja palju muud. Autor kasutas projektis mitmeid pakkette, kasutatud kolmanda osapoole pakettide kirjeldused on järgmistes peatükkides.

1. SimpleXMLRPCServer

RPC on protokoll, mis võimaldab arvutiprogrammil teostada funktsiooni erinevas aadressiruumis, näiteks teises serveris või arvutis, ilma, et programmeerijal oleks vaja kodeerida keerulisi suhtlusdetaile. RPC võimaldab programmil kutsuda protseduuri või meetodit teises süsteemis nagu oleks tegemist kohaliku funktsiooniga. Protokoll kasutab andmete saatmiseks XML vormistust. Autor kasutas RPC protokolli projektis väga tihti, enamus programmide vahelisest suhtlusest sai lahendatud läbi selle protokolli. Samuti suhtles UR robotkäsi peaarvutiga kasutades seda protokolli. [2] [3]

2. pymodbus

PyModbus on täielik modbus-i protokolli rakendus Phytonis, mis võimaldab, kasutades seda teeki, üles seada nii modbus-i serverit, kui ka klienti. Tegemist on kolmanda osapoole teegiga, millel on praeguseks juba üle saja erineva *Contributori* ehk kaasabistajat. Nii suur arv on võimalik ainult tänu sellele, et tegemist on avatud lähtekoodi projektiga, millele saab igaüks, kes tahab, panustada. Teeki kasutas autor, kui oli vaja üles seada Modbus-i server või kui oli vaja suhelda serveritega. Näiteks kogu pneumaatiliste silindrite juhtimine käis läbi manifoldi, mille juhtimine käis läbi modbus-i protokolli.

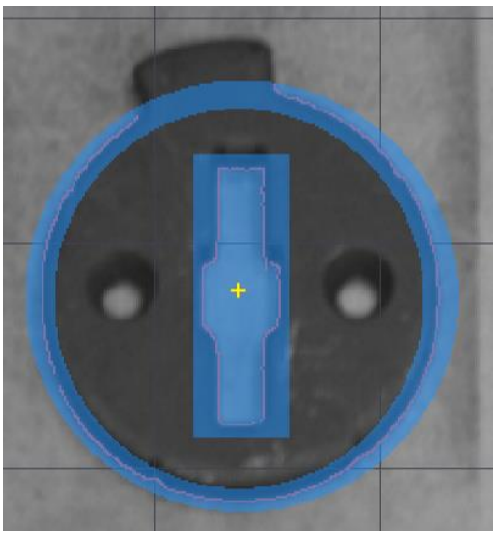
2.2 Aurora Vision Studio

Aurora Vision Studio on masinnägemise jaoks arendatud tarkvara, mis lihtsustab masinnägemise lahenduste arendamist. Tarkvaral on moderne kasutajaliides ning see ei nõua koodi kirjutamist programmide arendamiseks. Samuti on tarkvaral väga suur valik juba ette valmisdefineeritud filtreid, mida kasutades on võimalik programmeerijal peaaegu kõik masinnägemise probleemid lahendada ilma ise uusi filtreid arendamata. Järgnevates alampeatükkides kirjeldatakse keerulisemaid filtreid, mida kasutas autor kõige tihedamalt masinnägemise lahenduse koostamisel.

2.2.1 Mallide sobitamine

Mallide sobitamine (*ingl. Template matching*) on masinnägemise meetod, mis võimaldab tuvastada pildilt või kaadrilt eelmääratletud kujusid, targemad algoritmid suudavad ka määrata kujundi suuruse või nurga. Mallide sobitamise tehnoloogia võimaldab lahendada järgnevat vajadust: kahest pildist, millest üks on objekti võrdluspilt ning teine, mis on sisendpilt, leida üles sisendpildist kõik esinemised objektist kasutades tema võrdluspilti. Algoritm on üsna otsekohene ja lihtne üles seada, seetõttu on see muutunud üheks väga populaarseks meetodiks objektide lokaliseerimiseks. Projektis on kaks kohta, kus on vaja määrata objekti asukoht, et robot saaks neid haarata. Mõlemas juhtumis kasutas autor just mallide sobitamise meetodit, et see vajadus täita. [4]

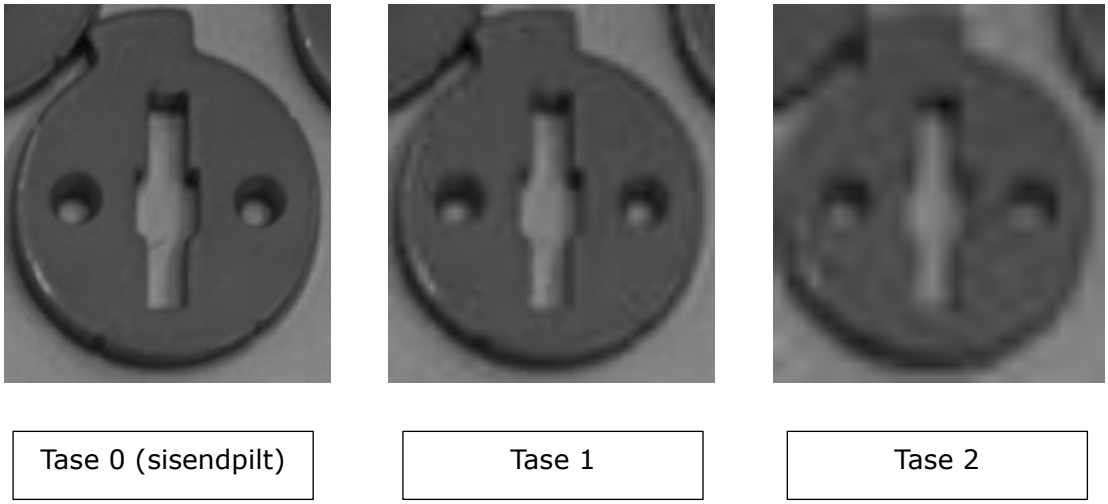
Täpsemalt kasutas autor alammeetodit nimega servapõhiline malli sobitus, mis optimeerib meetodit veelgi, kasutades vähem arvutusresursse kuna võrdleb ja otsib sisendpildist ainult detaili ääri, mitte ei proovi sobitada tervet detaili pilti. Meetodi kasutamiseks tuleb esiteks ette valmistada võrdluspilt, näiteks toodud projektis kasutatud OHZ detaili võrdlusmudel. (Joonis 5)



Joonis 5. OHZ detaili serva mudel

Võrdluspildil tuleb määrata detailil alad, kus asuvad ääred, mida hakatakse otsima sisendpildist. Joonis 5-l on märgitud see ala sinisega ning nurgad, mida programm tuvastab sellel alal, on märgitud punase joonega. Nendest äärtest koostatakse servamudel. Aurora Vision Studio tarkvara aitab selle mudeli koostamisel märkimisväärselt, kuna lubab neid alasid määrata kasutades pintslit, millega saab pildil lihtsalt värvida need alad, kust tahame ääri otsida. Servamudel koostatud, saab programm hakata sobitama seda sisendpildile, et otsida sealt detaile ja nende asukohta lokaliseerida. Kuid otse sisendpildilt detailide otsimine kasutades seda meetodit, on väga arvutusressursi nõudev ja aeglane.

Et optimeerida algoritmi, kasutatakse pildi püramiidi tasemeid. Püramiidi tasemed on seeria pilte, mille iga järgnev tase on eelmise pildi resolutsiooni alladiskreetmine. Praegusel juhul poole võrra väiksem resolutsioon. [4] Lõputöös kasutatud OHZ detaili püramiiditasemeid on näha Joonis 6.



Joonis 6. OHZ detaili püramiiditasemed

Tänu sellele tegevusele suurendame mallide sobitamise efektiivsust ja kiirust märkimisväärselt. Alustame sobitamist kõige kõrgemast tasemest ning kuna selle taseme resolutsioon on väga väike, siis kasutame selleks vähem aega ja arvutusresursse. Miinuseks on see, et saadud tulemus ei ole nii täpne, kuna detaili ääred on udused ning kindlat asukohta ei ole võimalik määrata. Pärast seda jätkame sobitamist sellest väiksemal tasemel, kuid otsime detaili ääri ainult selle koha ümbert, kust me need leidsime eelmisel tasemel. Sellega suurendame täpsust, otsides ääri võimalikult väikselt alalt ning mitte raisates aega, otsides neid kohtadest, kus neid kindlasti ei ole. Jätkame seda tsüklit, kuni jõuame välja kõige väiksemale tasemele, tase 0, milleks ongi sisendpilt. Kui tahame, võime lõpetada sobitamise varem suuremal tasemel, siis ignoreerime sisendpilti üldse ja suurendame algoritmi kiirust veelgi, kuid kaotame täpsuse. Kasutades püramiiditasemete

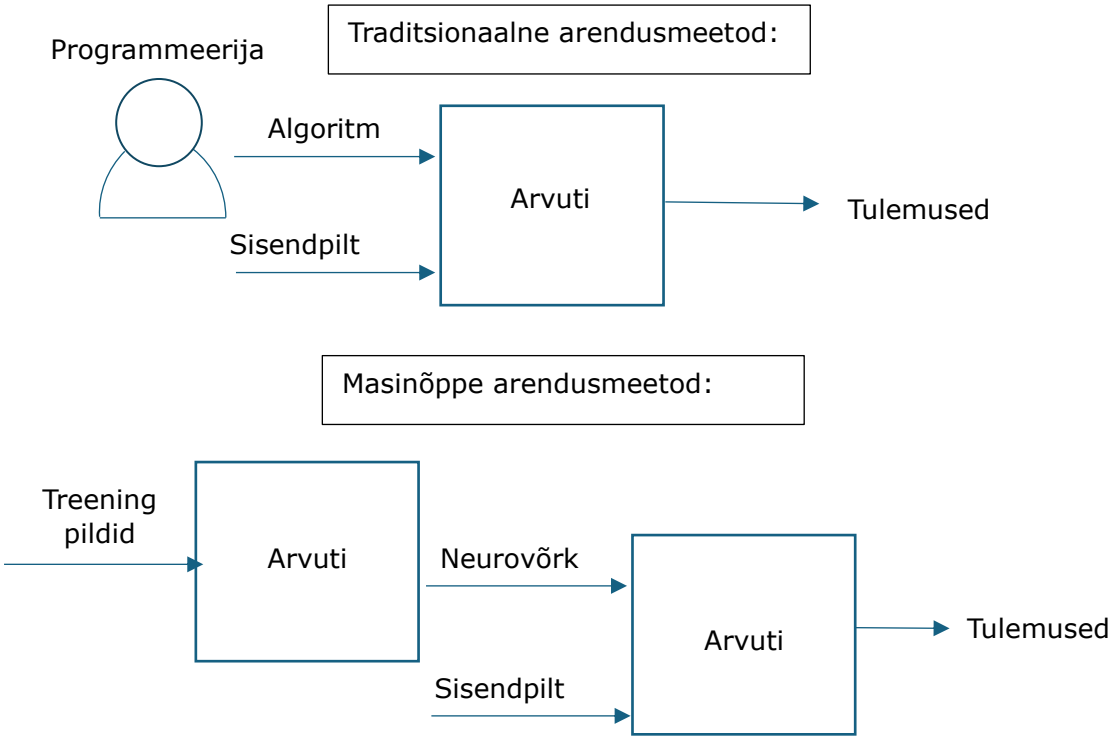
meetodit, saame määrata detaili asukoha ja nurga võimalikult täpselt, samas kasutades selleks võimalikult vähe ressursse [4].

2.2.2 Süvaõpe

Süvaõpe (*ingl. Deep Learning*) on revolutsiooniline meetod ja alternatiivne viis tavaliste algoritmide kirjutamisele. Tehnoloogia sai alguse juba 1949, kui Donald Hebb disainis esimese masinõppe mudeli, et simuleerida aju neuronite käitumist ning pärast seda kasutati seda 50-ndatel paljudel erinevatel eksperimentaalsetel kasutusaladel. Tehnoloogia muutus populaarseks alles 2000-nendatel ja leidis laialdaselt kasutust automaattootmises alates aastast 2010. [5]

Süvaõpe on samuti ka läbimurre arvutinägemises, võimaldades automaatselt lahendusi mitmesugustele pildianalüüsi rakendustele. See ületab traditsiooniliste algoritmide piiranguid eriti keerukate kujude või välimustega objektide kontrollimisel, kiirendades arendustsükklit. Samal ajal nõuab sügavõpe rohkem tähelepanu andmetöötlusele ja koolitusparameetritele, mis võtab enamiku rakenduse arendamise ajast. [6]

Võrreldes tavalise arendusmeetodiga näeb masinõppe algoritmide koostamine välja järgmiselt: (Joonis 7)



Joonis 7. Arendusmeetodite võrdlus [6]

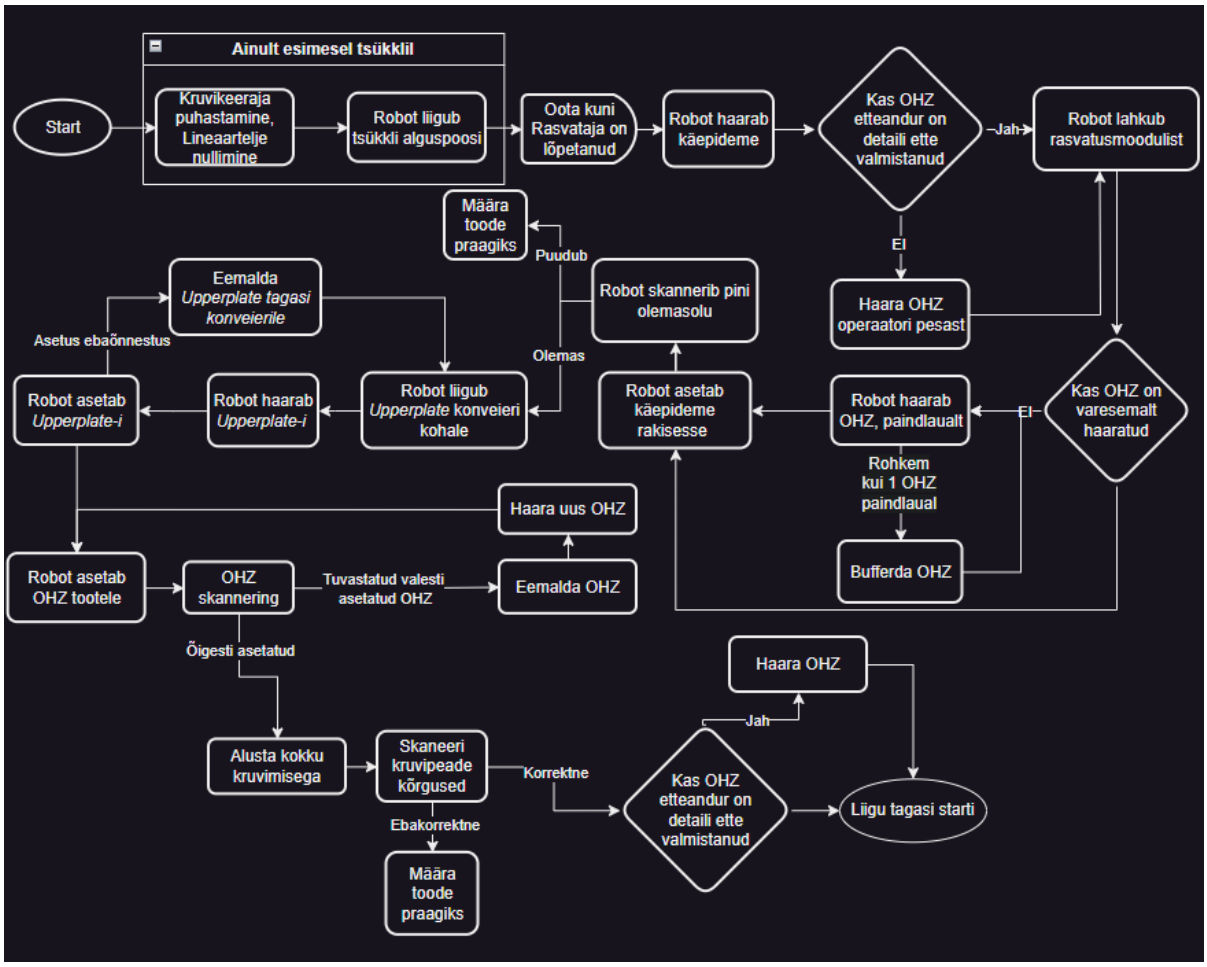
Masinõppe arendusmeetod erineb traditsionaalsest selle poolest, et algoritm, mis teeb otsuseid, ei ole koostatud arendaja poolt, vaid hoopis arvuti poolt. Arendaja ainult

valmistab ette suure koguse näiteid õigetest otsustest, mille põhjal arvuti koostab iseendale algoritmi.

Autor kasutas lõputöös süvaõppe tehnoloogiat OHZ detaili orientatsiooni määramiseks selle automaatetteanduris. Põhjalikumalt on seda kirjeldatud peatükis 6.3 Orientatsiooni määramine.

3 TOOTMISTSÜKLI KIRJELDUS

Tootmistsükkel kujutab endast nelja „võta ja aseta" ülesannet, kruvimist ja kvaliteedikontrolli. Joonis 8 on näha tootmistsükli kokkuvõtvat ülesannete järjekorra vooskeemi.



Joonis 8. Koostamisülesannete kokkuvõttev vooskeem

Robot võtab esimesena operaatori poolt asetatud käepideme rasvatamise moodulist. Rasvatamise moodul rasvatab käepideme sooned, kuhu asetatakse hiljem upperplate detail. Rasvatamise moodulis on ka lisa OHZ pesa, kuhu operaator peab asetama OHZ detaili enne, kui ta saab alustada rasvatamise tsüklit. See sai lisatud, kuna pärast testimist avastas autor, et OHZ etteandur ei suuda piisavalt kiiresti valmistada roboti jaoks detaili ette. Kuna tootmistsükkel oleks seetõttu pausile jäänud, siis nüüd sai robot haarata detaili sellest pesast siis, kui vajadus tekkis. Robot haarab OHZ detaili pesast juhul, kui OHZ etteandur ei ole veel jõudnud detaili ette valmistada vajalikuks hetkeks.

Järgmine etapp tavalises tsüklis on OHZ detaili haaramine selle etteandurist. Juhul, kui eelnevalt robot haaras OHZ detaili operaatori pesast, siis robot jätab selle sammu vahele.

Etteandur valmistab ette robotile haaravad detailid väristades paindlauda ja sellel olevaid detaile suvaliselt. Kui paindlaua kohal olev kaamera leiab mõne detaili pärast väristamist mis on roboti poolt haaratav, siis jääb OHZ etteandur ootama. Robot haarab need kasutades selleks mõeldud magnethaaratsit. Kuna tootmiskiirus on suur, siis autor arendas välja meetodi puhverdada lisadetailid, kui etteandur peaks leidma mitu haaravat detaili korraga. Etteanduri kõrval asub puhver, mis koosneb kahest pesast, kuhu on võimalik asetada roboti poolt lisa OHZ detailid ning vajadusel võtta need sealt. Lahendus võimaldas etteanduril saavutada suurem paralleelsus roboti tööga ja mitte oodata mõttetult detailide haaramist roboti poolt. Kui OHZ detail on haaratud paindlaualt või puhvrist, liigub robotikäsi edasi.

Robotkäsi asetab käepideme koostusrakisesse, kuid enne seda skaneerib laseriga, kas rakis on tühi. Autor programmeeris robotile sellise kontroll-loogika, kuna testimise käigus juhtus olukordi, kus pärast praaktoote eemaldamist jäi rakisesse mingisuguseid jääke, näiteks maha kukkunud OHZ või upperplate. Robot proovib eemaldada mainitud takistused, ebaõnnestumisel annab operaatoritele teate, et nad seda ise teeksid. Kui käepide on rakisesse asetatud, siis skaneerib robot, kas operaator on eelnevas positsioonis käepideme sisse korrektselt asetanud pinni. Juhul kui see puudub, määrab robot toote praagiks. Järgmisena liigub robotkäsi edasi upperplate etteanduri juurde.

Upperplate etteandur on konveier, kus kaamera otsib liikuvalt konveierilt detaile, mida robot saab haarata. Kui etteandur leiab detaili, mida robotkäsi on võimeline haarama, jääb konveier seisma ja ootab kuni robot selle eemaldab. Robotkäsi teeb pärast konveierilt haaramist lisakontrolli haaratsile kinnitatud optilise sensoriga, kas detail on korrektselt haaratud või üldse puudub. Kui detail puudub või on ebakorrektselt haaratsis, kukutab robot detaili tagasi konveierile ja jääb ootama uue detaili saabumist, mida haarata. Kui detail on haaratud, liigub robotkäsi tagasi rakise juurde, kus proovib asetada käepideme soontesse upperplate-i. Sooned käepideme sees on väiksed ja autor arendas upperplate-i asetamiseks sinna erilise liikumise. Nimelt pikendab robotkäsi kolbi, millele on haarats kinnitatud. Kolvi piirpositsioonis on autor seadistanud sensori täpselt niimoodi, et see annab signaali ainult siis, kui kolb on täielikult pikendatud. Kui upperplate on viltu või soontest natuke mööda, siis ei saa kolvil olev sensor enam signaali ja robotkäsi hakkab tegema pisikesi ringe. Kui upperplate kukub pärast seda soontesse sisse, siis saab kolvil asuv sensor signaali ja robotkäsi laseb detailist lahti.

OHZ detaili asetab robot täpselt upperplate-i peale. Kolvil, kuhu on kinnitatud OHZ magnetgripper, sai autori poolt seadistatud sensor piirpositsiooni, mis annab signaali ainult siis, kui kolb on täielikult pikendatud. Kui OHZ detail on korrektselt upperplate-i peal, siis saab sensor signaali, aga kui sensor ei saa signaali, siis teame, et mingi asi takistab OHZ

detaili asetust. Sellises olukorras eemaldab robot OHZ-i tagasi paindlauale ja proovib ühe korra uuesti uut OHZ-i haarata ja selle tootele asetada. Kui asetus peaks kaks korda järjest ebaõnnestuma, siis märgitakse toote praagiks. Kui OHZ on korrektselt asetatud, siis surutakse see rakises oleva giljotiiniga kinni.

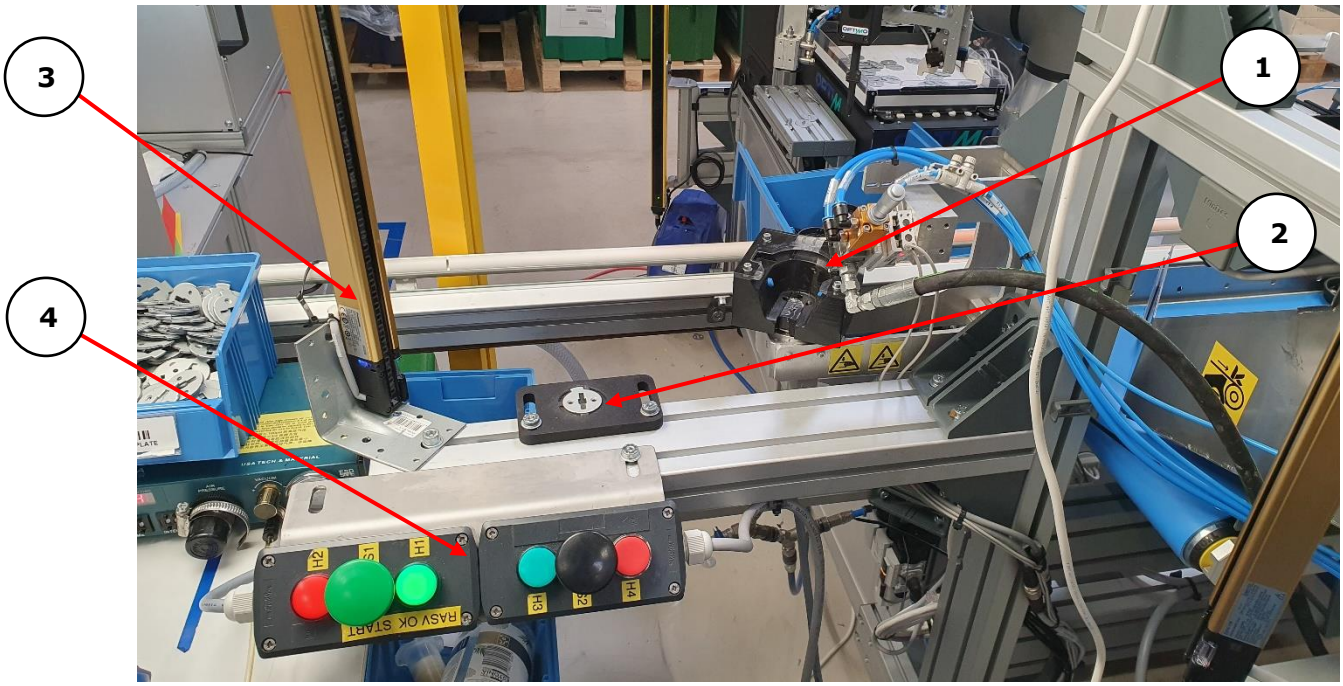
Järgmiseks alustab robot kruvimistsükliga, iga kruvi kruvitakse kinni täpselt kliendi poolt spetsifitseeritud jõumomendiga ja kui mõni kruvi jõumoment peaks olema ebanormaalselt suur või väike, siis märgitakse toode praagiks. Pärast kruvimist sooritab robotkäsi veel viimase kvaliteedikontrolli, kus skaneerib kruvipeade kõrgus võrreldes OHZ seibiga. Juhul kui peaks ilmnema ebanormaalsusi, siis märgistatakse toode praagina.

Tootmistsükli lõpetab lineaartelg, mis eemaldab valmistoote rakisest ja asetab selle konveierile, mis sõidutab selle edasi järgmisesse tootmisliini positsiooni. Lineaartelg sai lisatud automaatikapositsioonile selleks, et vabastada robotkäe tsüklist valmistoote eemaldamise ülesanne ja tänu sellele kiirendada keskmist tsükli aega.

Täpsemalt igale automaat-etteanduritele koostatud loogikast, keskenduvad järgmised peatükid.

4 KÄEPIDEME RASVATAMISE MOODUL

Kõige esimene etapp tootmisprogrammis on käepideme etteandur. Etteandur kujutab endast pesa, kuhu operaator asetab käepideme. Pärast nupu vajutust alustab moodul käepideme rasvatamise tsükliga. Rasvatamiseks oli kliendi enda poolt antud rasvakanister, mida kasutati varasemalt manuaalsel tootmisliinil. Rasvakanister tekitab rõhu, mida kasutades pressitakse rasv kanistrist välja ja läbi süstla täpselt käepideme soontesse.



Joonis 9. Käepideme rasvatamise moodul

(Joonis 9) Olevate mooduli komponentide seletus:

1. rasvatamise pesa;

Peamine osa moodulist, tegeleb käepideme rasvatamisega. Selleks, et soon käepidemes saaks ühtlaselt rasvatatud, on vaja käepidet keerata sujuvalt. Pesa, kuhu käepide operaatori poolt asetatakse, on kinnitatud Festo pöördtelje peale.

2. operaatori OHZ pesa;

Kuna automaatikapositsiooni testimise käigus ilmnes, et OHZ etteanduril ei ole võimalik ajaliselt iga 20 sekundi tagant üks roboti poolt haaratav detail ette valmistada, sai lisatud mooduli juurde pesa, kuhu operaator saab asetada OHZ detaili. Kui etteandur ei jõua valmistada detaili piisavalt kiiresti, saab robot selle haarata sealt tsükli alguses. Niimoodi ei jää tootmise ajal positsioon ootele.

3. turvakardinad;

Pesa, kuhu operaator asetab käepideme, et alustada selle rasvatamist, asub roboti tööalas. Selleks sai sinna paigaldatud turvakardinad, mis seiskavad roboti, kui nende vahele midagi satub. Turvakardinad aktiveeritakse ainult siis, kui robot on hetkel käepide haaramas.

4. nupumoodul.

Nupumoodulis kontrollitakse nii rasvataja kui ka terve positsiooni funktsioone. Kõige olulisem nupp on rasvatustsükli alustamise nupp. Kui operaator on käepideme pesasse sisestanud, siis vajutab ta rohelist nuppu, mis alustab rasvatamise tsüklit. Pöörtelg teeb liikumise teise positsiooni ja rasvataja lisab käepideme soontesse rasva. Pärast seda jääb pöörtelg ootama robotilt signaali, et käepide on haaratud ja siis pöörab ennast tagasi esimesse positsiooni, et alustada tsüklit uuesti. Lisaks on nupumoodulil ka kaks teist nuppu, must nupp saadab robotile *play* signaali. Nupp on vajalik, et käivitada positsioon uuesti, siis kui midagi peaks sattuma turvakardinate vahele. Selle puudumisel peaks operaator liikuma teisele poole tootmisliini, kus asub roboti kontroller, et seda sealt uuesti käivitada. Kolmas nupp ehk punane oli lisatud hiljem ja selle nupu vajutusel saadab positsioon signaali tootmisliini peakontrollerisse, et operaator avastas praaktoote. Funktsioon on vajalik korrektsete logide ja statistika säilitamiseks, mis on kliendi poolt nõutud. Kui operaator avastab, et eelmisest automaatikapositsioonist väljunud toode on praak, siis vajutab ta seda nuppu ja eemaldab toote liinilt. Sellega saame logidesse ära märkida, et tegemist oli praaktootega ja tootmisliin peab ühe lisatoote valmistama.

Pöördtelje positsioonide programmeerimine käis kasutades Festo enda programmi, sellele ligipääsemiseks pidi sisestama brauseriaknasse pöördtelje kontrolleri IP aadressi. Sellega pääses ligi kasutajaliidesele. Kasutajaliidesest sai määrata palju erinevaid parameetreid pöörteljel. Kõige olulisem autorile on määrata, et telg pöörab vastavalt IO signaalidele. Teine oluline asi on õpetada teljele liikumiste kiirused ja positsioonid (Joonis 10).

Record Table

Actual Position: 0.0000 U

< Jog neg. | > Jog pos.

No.	Positioning type	Position [U]		Velocity [U/s] max.=3.3333	Acceleration [U/s²] max.=432.9014	Torque [%] max.=100.0	
1	Positioning to absolute Position	0.0000	Teach Pos.	2.0000	200.0000	100.0	Move to Pos.
2	Positioning relative to actual Position	-1.0556	Teach Pos.	0.1528	100.0000	100.0	Move to Pos.
3	Positioning to absolute Position	-0.5889	Teach Pos.	1.0000	50.0000	100.0	Move to Pos.
4	Invalid	0.0000	Teach Pos.	0.0000	100.0000	100.0	Move to Pos.
5	Invalid	0.0000	Teach Pos.	0.0000	100.0000	100.0	Move to Pos.
6	Invalid	0.0000	Teach Pos.	0.0000	100.0000	100.0	Move to Pos.
7	Invalid	0.0000	Teach Pos.	0.0000	100.0000	100.0	Move to Pos.
			Download	Store			Stop

Joonis 10. Pöördtelje positsioonide tabel

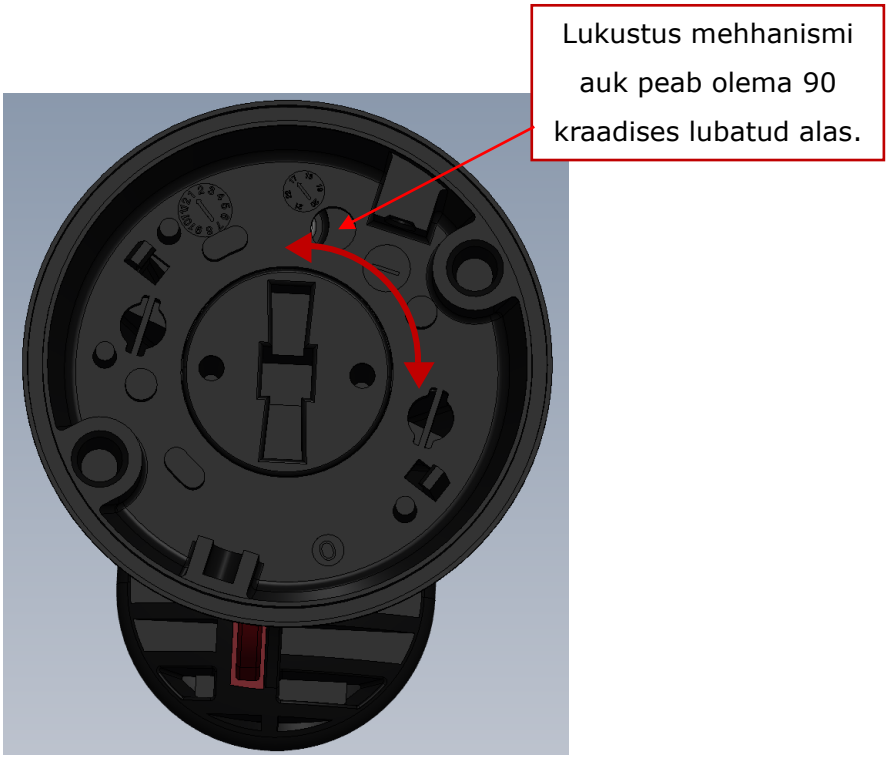
Pöördtelje positsioonide määramine käis kasutades Joonis 10 kujutatud tabelit. Mooduli tööks oli vaja määrata kolm positsiooni: esimene positsioon, kus operaator saab asetada käepideme pesasse, teine, mis oli täpselt ühe ringi võrra edasi keeratud, et pöördumise ajal rasvatada käepide ja kolmas positsioon, kust robotkäsi saaks haarata rasvatatud käepideme. Ülal oleval tabelis näeme neid positsioone ridadena. Igale positsioonile on vaja määrata sinna liikumise kiirus ja kiirendus. Esimesel ja teisel positsioonil ei olnud kiirus väga oluline ja autor määras kiirused nendel liikumistel võimalikult kiireks. Kolmandal positsioonil liikumine jäi natuke aeglasemaks, kuna siis on pesa sees käepide ja see ei tohi pöörlemise ajal pesast väljuda. Teine positsioon on täpselt ühe ringi võrra kaugemal esimesest. Selle ajal toimubki rasvatamise tsükkel, kus nõel lisab käepideme soonde rasva ja selle liikumiskiiruse pidi autor määrama selliseks, et terve soon käepidemes saab ühtlaselt rasvatatud.

5 UPPER PLATE DETAILI ETTEANDUR

Upperplate on järjekorras teine komponent, mida robot rakises komplekteerib. Upperplate-de etteandur on positsioonis kõige suurem. Etteandur moodustub kahest konveierist, üks horisontaalne ja teine diagonaalne ning kolust. Kaks konveierit moodustavad omavahel sõlme, mida mööda komponendid liiguvad. Horisontaalne konveier on mõeldud kaamera tööks ning alaks, kust robot saab probleemideta komponente haarata. Diagonaalne konveier viib detaile horisontaalse lõpust tagasi selle alguseni ning kukutab juppe muutes nende asetust ja orientatsiooni. Lisaks on etteanduri tööks ette nähtud ka kaks optilist sensorit ja kaamera. Upperplate on kahes erinevat versioonis, L-versioon ja J-versioon. J-versiooni toodetakse kõige sagedamini ja on selle peatüki põhiline fookus, L-versioon on eri versioon upperplate-ist, millele operaator sisestab manuaalselt veel ühe tihendi, L-versioonil ei ole oma automaatset etteandurit.

5.1 Probleemi kirjeldus

Robot peab asetama upperplate-i korralikult käepidemes olevale soontesse, kummitihend ülespoole. Kuna kummitihend on habras ja väga oluline valmiskomponendi tööks, ei tohi robot töö jooksul seda kahjustada. Upperplate-i asetades peab ka robot asetama selle õige nurga all, milleks on 90 kraadine lubatud ala, kuhu sisse peab jääma lukustusmehhanismi auk (Joonis 11).



Joonis 11. Upperplate asetuse juhend

Etteandurile on vaja arendada loogika, mis suudab leida õige nurga ja õige orientatsiooniga upperplatei, mida robot saab ilma singulaarsust tekkimata haarata.

Etteanduri asukoht suhtes robotiga on väga halva koha peal, roboti käe esimene liigend ei võimalda pöörata üle 363 kraadi [7, p. 130] ja konveieri suurem osa jääb sellest välja. Mis tähendab seda, et konveieri peal on ainult väga pisike ala, kust robot on tegelikult võimeline haarama.

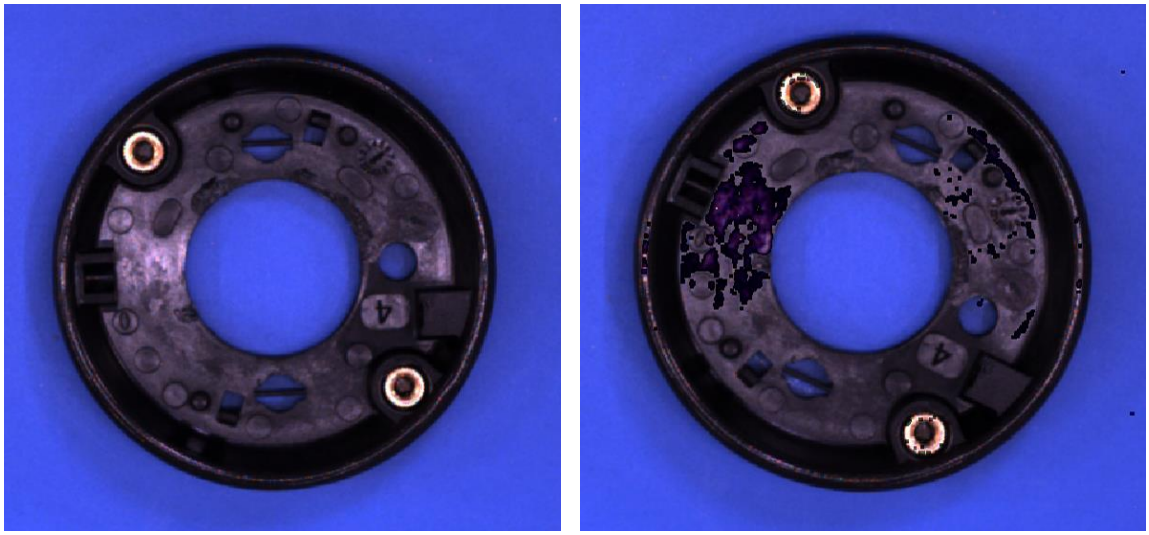
Lahenduseks arendas autor kaks eraldi töötavat programmi ning robotipoolse koodi. Esimene programm ehk kaameraprogramm, kasutab masinnägemist ja määrab kindlaks konveieril olevate upperplatide orientatsiooni nurga ja asukoha. Teine programm on Phytoni programm, mis töötleb kaameraprogrammilt saadetud andmeid ja määrab kindlaks, millise nurga all ja millisest asukohast peab robot detaili haarama, konverteerides asukoha kaamera koordinaatteljestikust roboti koordinaatteljestikku. Samaaegselt kontrollib programm ka konveierite ning kolu hopperduse loogikat. Robotipoolne kood juhib UR robotit, mis haarab konveierilt detaili ja asetab selle käepideme peale rakises, pöörates seda vajalikul mahul, et lukustusmehhanismi auk jääks lubatud alasse.

5.2 Asukoha, nurga ja orientatsiooni määramine

Kaameraprogramm on arendatud kasutades Aurora Vision Studio-t ning koosneb kahest osast, pildi saamise ja ettevalmistamise filter ning detaili asukoha ja nurga määramine.

Kuna detaili asukoha ja nurga määramine käib kasutades serva sobitamise meetodit, siis segavad programmi tööd igasugused läiked detailil, kuna need tekitavad detaili pildile ääred kohtadesse, kus neid tegelikult ei ole. Ääreks nimetatakse kohta pildil, kus pikslite vahel muutub nende toon või heledus üle piirmäära. Arenduse alguses, kus autor ei olnud veel programmeerinud lahendust läigete eemaldamiseks, juhtus tihti selline olukord, kus kaameraprogramm arvas, et detaili nurk konveieri peal on umbes 180kraadi valet pidi. See juhtus seetõttu, et detailis olevale lukustusmehhanismi augule, mida kasutatakse nurga määramiseks, tekkisid sarnased läiked täpselt teisel pool detaili (Joonis 12).

Autor lahendas selle probleemi arendades läikefiltri, mis leiab kaadrilt üles kõik pikslid, mille RGB värviväärtused langesid teatud piiridesse, ning vähendab nende pikslite heledust 50 ühiku võrra (värviväärtused piirduvad vahemikku 0 - 255). Pärast seda muutub läike värv märgatavalt tumedamaks ning ei tekita enam nn „kummitus-servi“ (Joonis 12).



Joonis 12. Upperplate sisendpilt(vasakul) ja läikefilter(paremal)

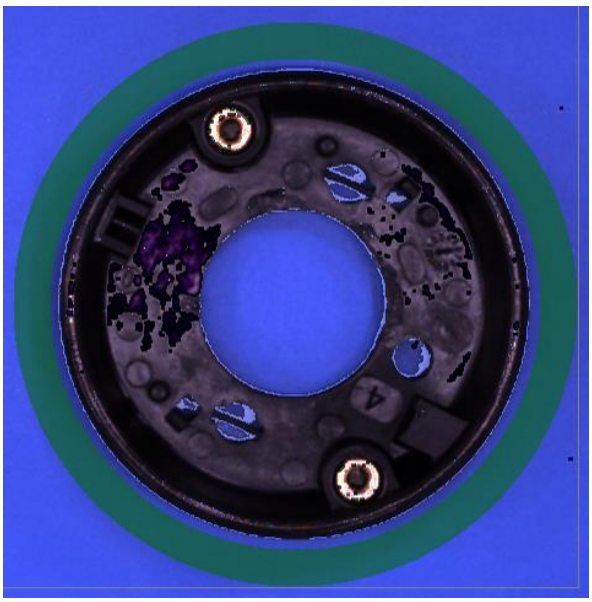
Detailil nurga ja asukoha määramiseks kasutas autor auke, mis on detailil. Täpsemalt kolme auku, mis on näha joonisel 12. Kuid olenevalt ajast ja ilmast on valgustase tase tehases erinev ja mõnikord ei paista need augud välja kaamerale. Selle probleemi lahendus on sarnane läikeprobleemi lahendamisele, programm leiab pildilt pikslid, mis on teatud RGB väärtustega. Kuna konveier, mille peal detailid liiguvad, on sinine, siis saab otsida kaadrilt tumedaid piksleid, mille B ehk sinine väärtus on märkimisväärselt kõrgem teistest piksli värvikomponentidest. Sellega leiab kaadrilt täpselt need pikslid, mis on detaili aukudes. Muutes nende pikslite ereduse suuremaks, tulevad esile detaili ääred (Joonis 13).

Joonis 13. Upperplate aukude filter

Pärast seda on kaadri ettevalmistus tehtud ja kaader saadetakse edasi programmi teise etappi, kus määratakse detaili asukoht ja nurk.

Asukoht ja nurk sai määratud mallide sobitamis meetodil (2.2.1 Mallide sobitamine). Püramiidi tasemete vahemik servamudelil jäi miinimum 2 ja maksimum 4 vahele, detail on suur ja paistab väga eredalt välja sinisel konveieril, mis võimaldas sellise kõrge tasemete vahemiku. Kõrgem tase ei kasutanud ka palju arvutusressursse. Autor konfigureeris filtri niimoodi, et see tuvastab kaadrilt ainult ühe detaili. Kui kaadril peaks olema mitu detaili, siis tuvastab programm ainult selle detaili, mille sobitusprotsent on kõige kõrgem. Orientatsiooni määramiseks ei pidanud arendama ühtegi eraldi filtrit, kuna valepidi oleva detaili väljanägemine on piisavalt erinev ja mallide sobitamise filter lihtsalt ei tuvasta tagurpidi olevat detaili.

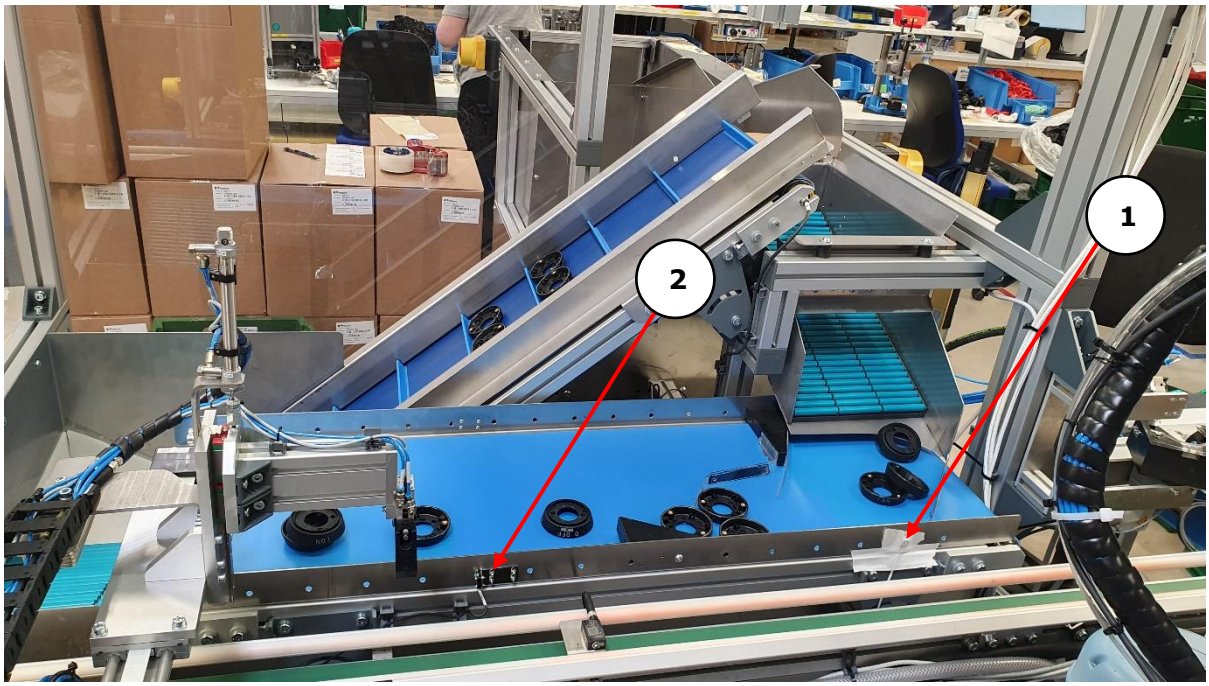
Pärast testimist, mis seisnes roboti ööseks tööle jätmist ning kus robot haaras konveierilt detaili ja kukutas selle tagasi selle peale, osutus haaramise täpsuseks umbes 95 protsenti. See on halb tulemus, mingisugune viga programmis takistas detaili korrektset haaramist. Veaks osutuks see, et kui kaks detaili täpselt üksteist kõrval paiknevad, siis tuvastatakse üks nendest korrektselt aga kui robot läheb seda detaili haarama, siis hakkab kõrval paiknev detail seda segama. Roboti haarats jääb kõrval asetuva detaili taha kinni ja ei ole võimeline enam saama head haaret. Probleemi lahendamiseks arendas autor lisaks uue filtri, nn „Kliirens filtri" (Joonis 14). Filter määrab detaili ümber kujutleva ringikujulise ala, mis on joonis 14-l välja toodud rohelise värviga. Filter analüüsib selle ala sees olevaid piksleid ja kui pikslite keskmine heleduse väärtus on liiga madal, siis saab eeldada, et kuskil detaili juures asub teine detail ja esimest tuleb ignoreerida.



Joonis 14. Upperplate kliirens filter

5.3 Etteanduri kontroller

Teine osa Upperplatei mooduli positsioonist on etteanduri kontrollerprogramm, mis juhib konveierite ja kolu hopperduse tööd ning valmistab detaili roboti jaoks ette, konverteerides asukoha kaamera koordinaatteljestikust robotiteljestikku ja arvutab, mis nurga alt robot peab detaili haarama. Joonis 15 on illustratsioon etteandurist, tööks olulised sensorid on nummerdatud.



Joonis 15. Upperplate etteandur

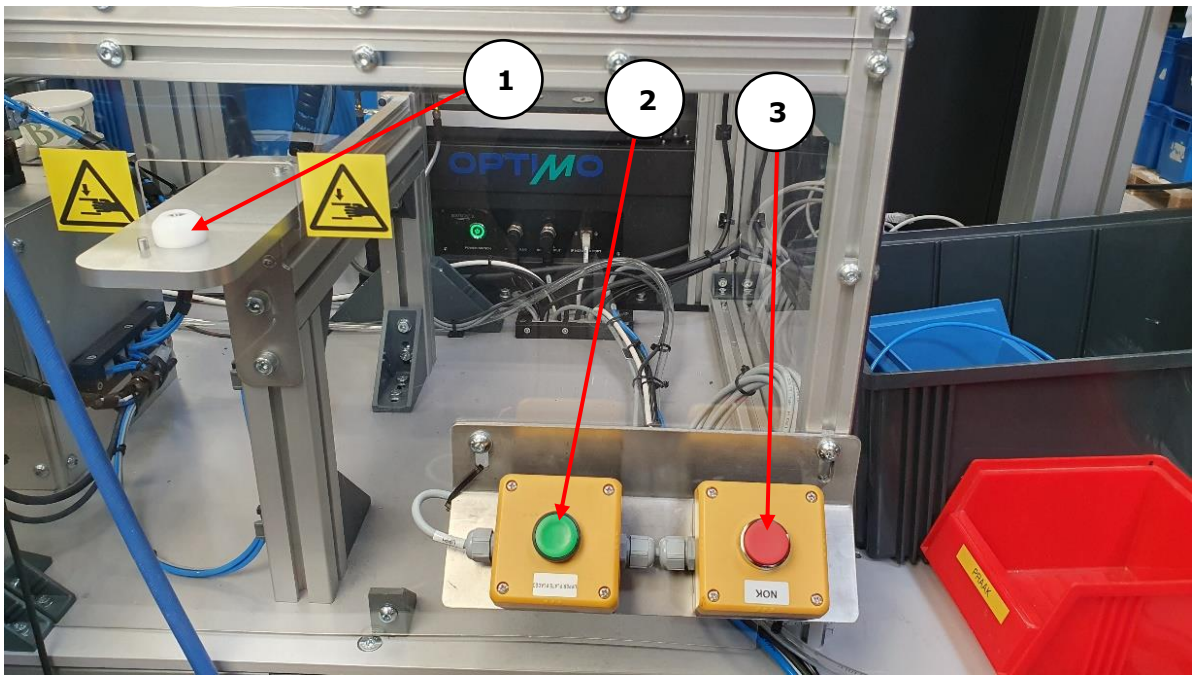
Etteandurite konveierite loogika sai arendatud autori poolt võimalikult lihtne. Diagonaalne konveier töötab kogu aeg, aga seiskub ainult kolmel juhul: Siis kui sensor nr 1. saab signaali, et tekitada sellelt kukkuvatele detailide vahele distantsi; siis kui horisontaalne konveier seiskub, et protsessida kaamera poolt nähtud detaili ja oodata, et robot selle haaraks ja siis kui kolu doseerib uusi detaile juurde. Horisontaalne konveier seiskub ainult siis, kui sensor nr. 2 saab signaali, pärast seda on upperplate detail täpselt kaadris. Kolu doseerib uusi detaile juurde, kui sensor nr. 1 ei ole kuue sekundi jooksul saanud kordagi signaali. Kui pärast doseerimist ei saa sensor nr. 1 paari sekundi jooksul signaali, siis saadab etteandur signaali peaarvutisse, et kolu on vaja täita.

Robot on võimeline haarama detaili paindlaualt ainult teatud nurga vahemikus. Kui robot proovib haarata detaili sellest vahemikust väljas, siis võib sattuda robotkäsi singulaarsusesse või anda hoobi iseendale, kuna keerab otsaefektorit liiga palju. Pärast detaili rakisesse asetamist on robotkäsi võimeline keerama detaili ainult teatud mahus enne iseendale hoobi andmisel. Autor arendas programmi, mis arvutab välja, milliste

nurkade all oleva detaili on robot võimeline haarama. *Joonis 11* nähtav 90 kraadine ala, mille piires robot suudab detaili asetada. Autor leidis võimaluse seda ala suurendada. Esiteks määras autor selle keskpositsiooni 0 asukohaks. Mis tähendab, et algne nurkade piirväärtus on -+45°. Sellele liidab autor vahemiku, mis robotkäsi on võimeline detaili keerama rakises, see suurendab ala -+82° suuruseks. Järgnevalt liidab autor veel vahemiku, mis robotkäsi on võimeline etteanduri konveieri pinnal ennast keerama. Lõpptulemuseks sai autor vahemiku -+126°, mis tähendab, et 70% detailidest, mis etteanduril kaamera tuvastab, on roboti poolt haaratavad.

5.4 L-versioon

L-versioon on variatsioon upperplate-ist, kuhu operaator sisestab lisatihendi. Selle variatsiooniga tootmisel ei kasutata automaatetteandurit, kuna tihend, mille operaator sisestab, võib muidu sealt välja kukkuda. Mehaanikud olid projekteerinud selle variatsiooni jaoks eraldi pneumaatilise lineaartelje, mille peale saab asetada upperplate-i ja pärast nupuvajutust liigub lineaartelg positsiooni sisse, kust robot saab selle haarata. Autor programmeeris roboti iga kord, kui algab uus tsükkel, kontrollima peaarvutist milline Upperplate versioon hetkel tootmisel on ja siis vastavalt sellele kasutama kas L-versiooni lineaartelge või automaat-etteandurit. Et vähendada müratugevust ja elektrienergiat, siis lülitab J-versiooni etteandur ennast ise välja, kui hetkel tootmises kasutatakse L-versiooni.



Joonis 16. L-versiooni moodul

Joonis 16. Välja toodud märgised L-versiooni moodulil:

1. L-Versiooni süstik;

Operaator asetab L-versiooni upperplate-i, lineaartelje peal asuvale süstikule. Süstikul on upperplatei õigeks asetuseks lisatud metallist pinn, mis kinnitab selle õige asetuse operaatori poolt.

2. lineaartelje käivitus-nupp;

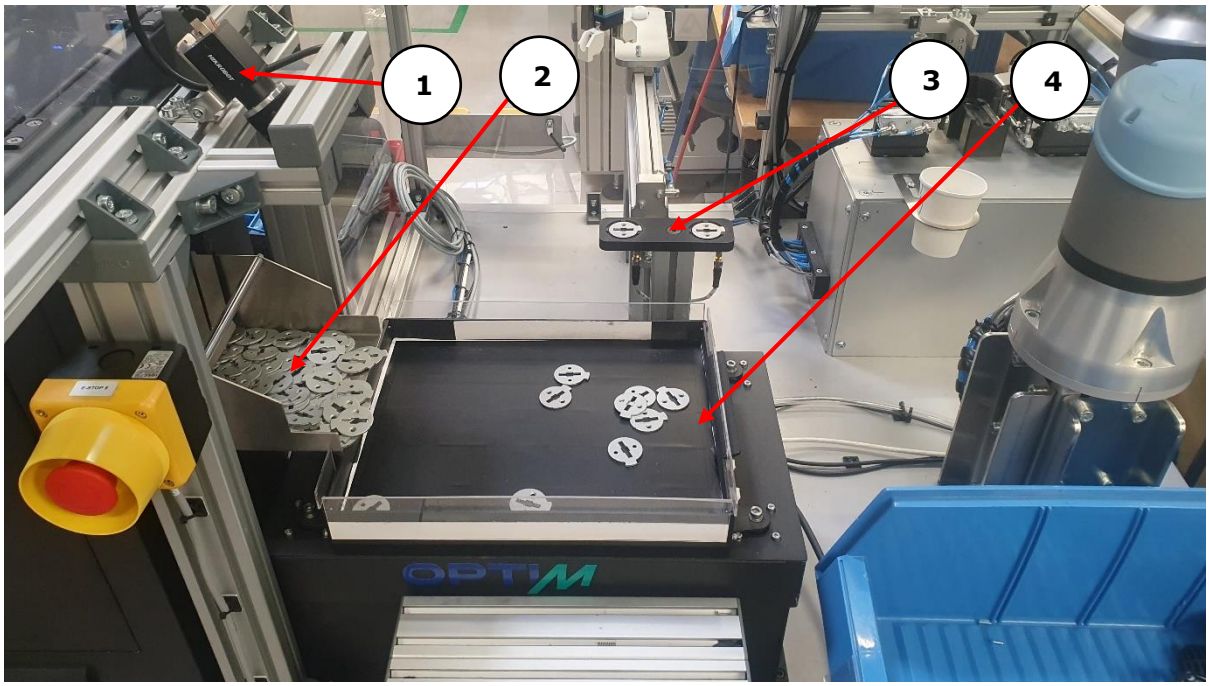
Pärast L-versiooni upperplate asetust vajutab operaator rohelist nuppu, mis saadab süstiku positsiooni sisse, et robot saaks sellelt haarata detaili.

3. praaktoote nupp.

Nagu varasemalt näidatud rasvatamise moodulil, on sellele nuppu moodulile ka lisatud praaktoote nupp. Operaator vajutab nuppu, kui avastab praaktoote, mis tuleb kolmandast positsioonist välja. See saadab signaali peaarvutile, mis teeb logidesse märkme.

6 OHZ DETAILI ETTEANDUR

Kolmas etapp koostustsüklil on OHZ detail asetus, UR robotkäe loogikapuu poolt vaadates, kuid detaili enda ettevalmistus toimub paralleelse tegevusena etteanduris. Ohz detaile on kolmes erinevas variatsioonis, kaks erinevat kahe kruviauguga variatsiooni ja nelja kruviauguga versioon. Detaili etteanduriks oli muretsetud vibreeriva pinnaga paindlik söötmissüsteem. (Joonis 17)



Joonis 17. OHZ detaili etteandur

Joonis 17 Välja toodud märgised:

1. kaamera;

Kaamera on kinnitatud paindlaua kohale ja seadistatud nurga alla, mis näeks tervet paindlauda.

2. OHZ detailide kolu;

Detailide kolu hoiab suuremas koguses OHZ detaile ja doseerib neid paindlauale peale automaatselt, kui nende kogus peaks vähenema.

3. OHZ detailide puhver;

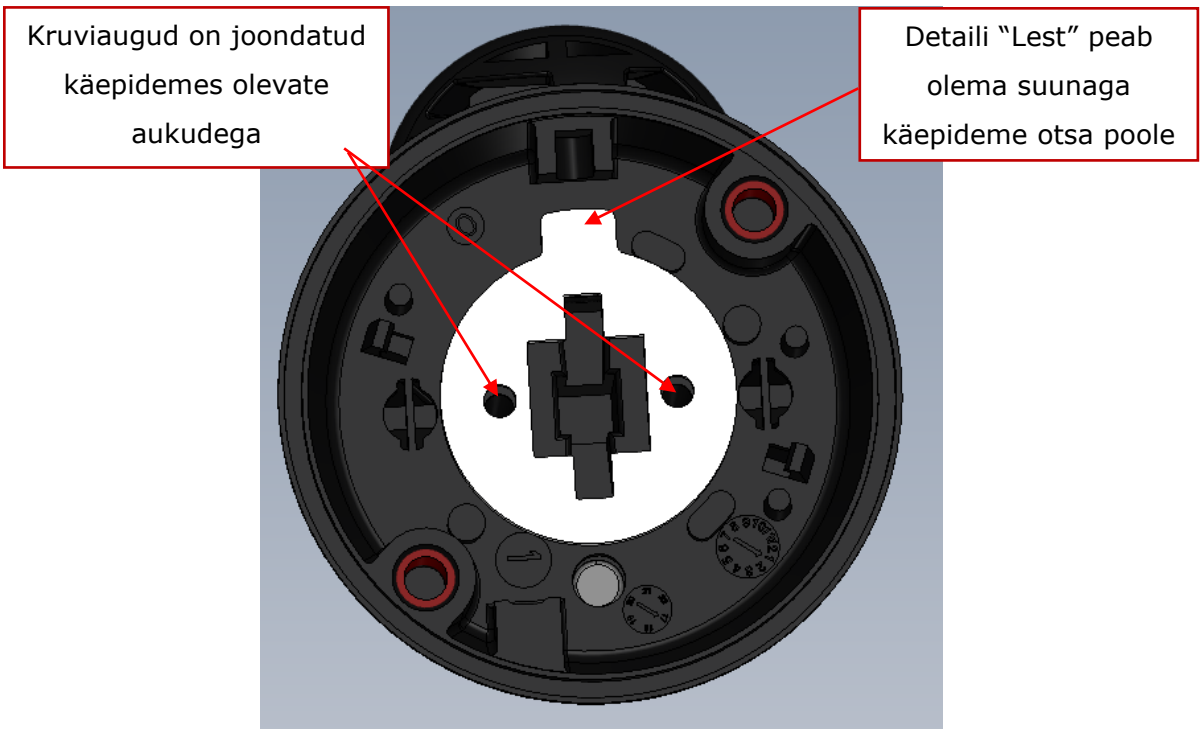
OHZ detailide puhver, kuhu robot saab asetada ja haarata OHZ detaile vastavalt vajadusele. Detail oli vajalik, et kiirendada keskmist tsükli aja kiirust.

4. paindlaud.

Paindlaud, mis väristab oma pinda ja selle peal asuvaid OHZ detaile, proovides neid teha robotile haaratavaks.

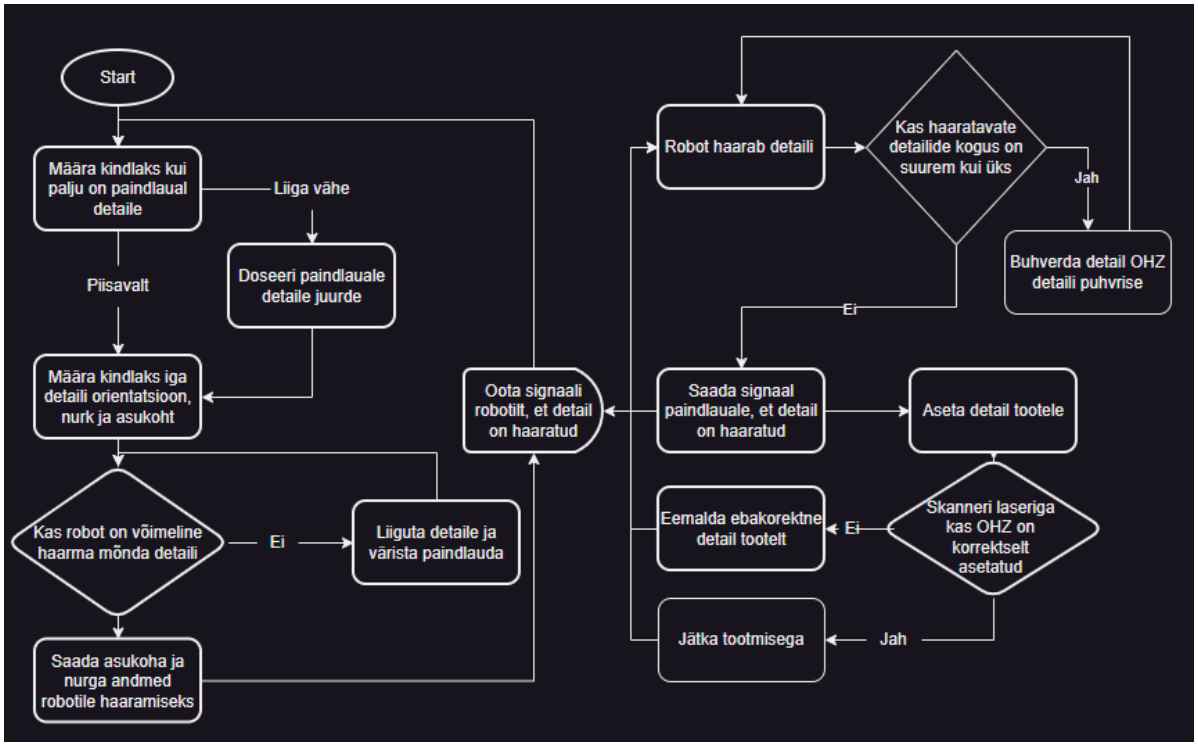
6.1 Probleemi kirjeldus

Ülesanne, mida robot OHZ detailiga koostama peab, on oma olemuselt lihtne. Nimelt on OHZ vaja asetada täpselt ja korrektselt eelnevas etapis saadud Upperplate-i peale, detaili "lest" suunaga käepideme otsa poole (Joonis 18). Kuna kruviaukudele on detaili ühele poolele sisse lõigatud sooned, mis peavad olema suunaga üles, siis peab detaili orientatsioon ka olema määratud etteanduris. Kui OHZ on asetatud viltu, siis järgmine etapp, mis hõlmab kõikide detailide kokku kruvimist, ebaõnnestub ja rikub ära terve toote. Praagi koguse vähendamiseks oli vaja arendada kontroll, kus mõõdetakse detaili korrektset asetust. Vale asetuse korral eemaldatakse detail ja proovitakse uuega uuesti.



Joonis 18. OHZ detaili korrektne asetus tootel

Kõik see tähendab, et detaili ettevalmistus, kus tuli kindlaks määrata detaili orientatsioon, nurk ning kas robot on üldse võimeline haarama seda detaili paindlaualt, võttis suurema osa arendusprotsessi ajast. Probleemi lahenduseks arendas autor välja süsteemi, mille kokkuvõttev diagramm on järgnev (Joonis 19).



Joonis 19. OHZ detaili süsteemi vooskeem

Joonis 19 vooskeemis on näha autori poolt arendatud süsteemi vooskeemi. Vooskeem koosneb kahest osast: automaat-etteanduri loogika (vasakul) ja UR robotkäe loogika (paremal). UR robotkäe poolne loogika ei ole lineaarne, robotkäsi haarab detaili paindlaualt ja kui haaratavaid detaile peaks olema rohkem kui üks, siis puhverdab robotkäsi ühe nendest Joonis 17-l nähtavale puhvrisse. Kui puhver on juba täis, siis ignoreerib robotkäsi teist detaili. Robotkäsi võtab OHZ detaili puhvrist siis, kui automaat-etteandur ei ole jõudnud ette valmistada detaili piisavalt kiiresti. Pärast seda jätkab robotkäsi haaratud detailiga tootmist ning kui kvaliteedikontrolli ajal avastab robotkäsi, et OHZ detail on asetatud viltu, siis eemaldab see selle tootelt ja pärast uue detaili haaramist proovib uuesti. Automaat-etteanduri loogika on täpsemalt lahti seletatud järgnevates peatükkides.

6.2 Asukoha ja nurga määramine

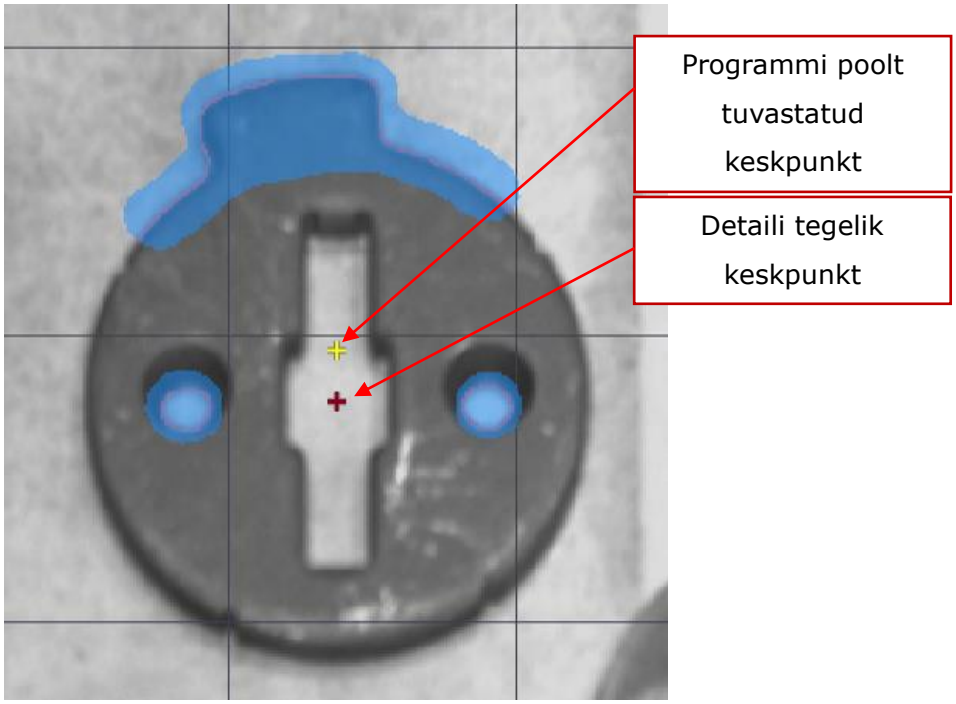
Detaili asukoha ja nurga määramine toimub kaameraprogrammis ja kasutades varem kirjeldatud mallide sobitamise tehnoloogiat (*2.2.1 Mallide sobitamine*).

Autor koostas OHZ detaili servamudeli ning pärast testimist leidis, et kõige parem püramiidi tasemete vahemik täpsuse ja kiiruse jaoks oli miinimum 1 ja maksimum 3. See on väiksem võrreldes Upperplate mudeliga ning põhjuseks on see, et OHZ detailide kogus paindlaual oli märkimisväärselt suurem ja detaili välimus on väga pisikeste ning raskesti eristatavate tunnustega. Kui püramiidi taseme maksimum oleks olnud suurem, siis ei suudaks programm suurtematel tasemetel enam servi täpselt tuvastada. Tulemus oli hea,

asukoht sai määratud piisavalt täpselt ning robot sai detaili haaramisega hakkama ilma probleemita. Kahuks isegi kui see filter oli asukoha määramisel täpne, siis detaili nurgaga mitte nii väga. Tihti esines olukordi, kus tuvastatud nurk oli täpselt 180 kraadi pööratud või täiesti vale. Koostatud servamudel oli seega väga efektiivne asukoha määramiseks, nurga määramiseks aga mitte. Mingisuguse kompromissi leidmine oleks võtnud liiga palju aega ja autor otsustas, et kõige parem lahendus on lihtsalt koostada teine servamudel, mille ainus eesmärk oli nurga tuvastamine.

Asukoht määratud, võttis programm iga detaili asukoha ja joonistas nende ümber kujutleva kasti, suurus oli piisavalt suur, et terve detail selle sisse mahutada. Autor edastas selle ruudu sees oleva pildi järgmisesse filtrisse, mille eesmärgiks oli detailide nurga määramine, kus kasutati mallide sobitamistehnikat, aga seekord ei otsinud programm ääri tervelt kaadrilt, vaid iga detaili enda kujutlevalt ruudult. Tänu sellele oli protsess kordades kiirem ja ei koormanud liiga palju arvutit. Enam ei tekkinud varem mainitud vigasid nurga määramises ja probleem sai lahendatud.

Pärast põhjalikku testimist, kus robot jäi ööseks paindlaualt detaile haarama ja kasti asetama, osutus haaramistäpsuseks 93%. See on hea, aga mitte piisav, kuskil esines viga, mis määras detaili asukohta või nurga valesti. Pärast edasist testimist avastas autor, et probleem esines detaili asukoha määramises, kuid mitte otseselt selle tuvastamises. Nimelt koostatud servamudel määras detaili keskpunkti ebatäpselt. Põhjuseks oli tarkvara enda viga, mis eeldas, et detaili kesk asukoht oli selle servamudeli geomeetriline keskmine, mis on enamus detailidel õige aga mitte konkreetsel detailil, kuna OHZ-l olev „lest" rikub selle ära. (*Joonis 20.* OHZ detaili servamudeli viga)



Joonis 20. OHZ detaili servamudeli viga

Nihe sõltub detaili nurgast, lihtsalt Y-koordinaadi allapoole nihutamist ei ole võimalik teha, kuna kui detail oleks teise nurga all, siis muutuks nihutamise suund. Probleemi saab lahendada kergesti kasutades trigonomeetriliseid funktsioone. Esiteks konverteerime eelnevalt saadud nurga radiaanidesse. Siis arvutame nihke horisontaal- ja vertikaalkomponentid kasutades siinust ja koosinust. Saadud komponendid lahutame algsetest koordinaatidest. Lõpptulemuseks saame korrigeeritud koordinaadid, mis on nihutatud vastavalt detaili nurgale. Arvutuskäigust koostatud valem (*Valem 1*) on järgnev:

$$x' = x - d \cdot cos(\theta) \quad (1)$$

$$y' = y - d \cdot sin(\theta)$$

kus x’ ja y’ - objekti uued koordinaadid pärast nihet, pikslid;

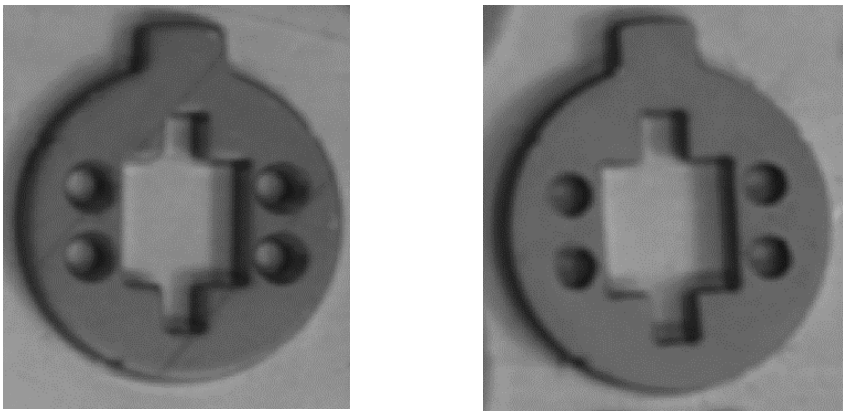
θ - pöördenurk, rad;

d - nihutamise kaugus, pikslid.

Valem 1-te autori poolt andmed sisse pannes saab asukoha viga korrigeeritud ja roboti haaramistäpsus tõusis aktsepteeritavasse suurusesse. Autor koostas valemi põhjal uue filtri Aurora Vision Studios mis võttis sisendiks detaili koordinaadid ja nurga, väljundiks arvutab korrigeeritud OHZ detaili asukoha. Detaili nurga ja asukoha määramise probleem sai autori poolt lahendatud. Järgmiseks oli vaja määrata detaili orientatsioon.

6.3 Orientatsiooni määramine

Orientatsiooni määramine detailil otsustus kõige keerukamaks probleemiks kogu automaatika positsioonis. Ainus võimalus vahet teha õiget- ja valepidi oleval detailil on see, kas kruvifaasid on näha või mitte, kas faas detaili äärel on teise raadiusega või pinnatöötlus on natuke erinev. Väga väikesed vahed, millel programmikoodil on raske vahet teha. (Joonis 21)



Joonis 21. OHZ õigetpidi(vasakul) ja valepidi(paremal)

Probleemi lahendamiseks testis autor palju erinevaid meetodeid, proovides tuvastada kruviauke ja siis lugeda nende suurust või tooni, et tuvastada faase. See lahendus ei töötanud, kuna nende vahe on liiga pisike, et saada õige ja järjepidev tulemus. Kõik pinnatöötuse määramise lahendused ei töötanud, kuna valgustase tehases muutus liiga palju ja see andis ebatäpseid tulemusi. Lõpuks leidis autor, et kõige parem ja järjepidevam lahendus oleks arendada masinõppe mudel ja kasutada süvaõppe meetodeid. (*2.2.2 Süvaõpe*)

Arendusprotsess algas sellest, et autor kogus suure koguse pilte detailidest - õigetpidi, tagurpidi, erinevates asukohtades, erinevate valgustasemetega, erinevate obstruksioonidega ja osad defektiga tooted. Arvutiteaduses on olemas väljend „*Garbage in, garbage out*" (prügi sisse, prügi välja), mis iseloomustab seda, et ükskõik kui hästi on algoritm arendatud, kui sisendandmed on halva kvaliteediga, siis tulemus on ka alati halb. Autori olukorras tähendas see seda, et pidi koguma võimalikult palju ja võimalikult kvaliteetseid pilte detailidest. Ilma seda tegemata neurovõrk, mida autor arendas, ei suudaks määrata täpselt detailide orientatsiooni. Autor otsustas, et treeningandmete kogus peab olema võimalikult suur. Kogus osutus umbes 800 pilti igast variatsioonist ehk kokku 2400 pilti. Piltide kogus oli nii massiivne, et manuaalselt neid pildistada ei oleks praktiliselt võimalik. Selleks arendas autor piltide kogumiseks iseseisva programmi, mille ainus ülesanne on leida paindlaualt kõik detailid kasutades eelnevas peatükis mainitud

meetodeid. Programm kärbib pildilt iga detaili eraldi välja ja keerab selle niimoodi, et detaili „lest" oleks alati ühtepidi ning salvestab pildi. Programm teeb seda iga detaili kohta, mille leiab paindlaualt ja seejärel väristab paindlauda, muutes detailide asukohta ja orientatsiooni.

Järgmiseks oli vaja määrata igal pildil, kas tegemist on õigetpidi või valepidi oleva detailiga. Selle protsessi tegemiseks ei olnud ühtegi muud võimalust, kui manuaalselt iga pilt eraldi üle vaadata ja märgistada. Protsess võttis aega terve tööpäeva, kuid autor ei teinud seda kõike ühekorraga, kuna tegemist oli väga rutiinse tööga.

Järgmine etapp on neurovõrgu treenimine, seda teeb arvuti ise ja see võtab tavaliselt kaua aega, keskmiseks ajakuluks iga mudeli kohta oli umbes 40 minutit. Protsessi oleks saanud märkimisväärselt kiirendada, kui arvutis, kus toimus treenimine, oleks kasutatud videokaarti, kahjuks see võimalus puudus. Neurovõrgul, mida treeniti, on võimalik määrata väga palju erinevaid parameetreid, kõige olulisemad parameetrid autori töös on neurovõrgu sügavus ja treeningandmete jagamine.

Neurovõrgu sügavus on abstraktne väärtus, mis väljendab kui palju mäluressurse on neurovõrgul lubatud kasutada. Mida suurem on paremeeter, seda väiksematest erinevustest klasside vahel on neurovõrk võimeline vahet tegema. Kahjuks tähendab see ka seda, et mida sügavam neurovõrk on, seda aeglasem on selle treenimine ja pärast jooksutamine. Aurora Vision Studio lubab määrata neurovõrgu sügavust vahemikus 1-10. OHZ detaili orientatsiooni määramiseks kasutas autor kõige suuremat ehk tase 10 sügavust. Väiksemate väärtsutega ei suutnud mudel enam järjepidevalt kindlaks teha detaili orientatsiooni.

Treeningandmeid saab jagada kolme erinevasse rühma - õppimisandmed, valideerimisandmed ja testimisandmed.

Õppimisandmed on näidete või proovide kogumid, mida kasutatakse masinõppe mudeli õpetamiseks. Mudel kasutab õppimisandmestikku, et mõista andmete mustrite ja suhete olemust, õppides seeläbi tegema ennustusi või otsuseid. Neid kasutatakse esimeses etapis, mille tulemusena valmib treenitud mudel, aga mille tegelikku täpsust teha õigeid otsuseid ei teata.

Teise etapina hinnatakse treenitut mudelit. Valideerimisandmed sisaldavad erinevaid pilte, et hinnata treenitud masinõppemudeleid. Sellel etapil on endiselt võimalik mudelit häälestada ja kontrollida. Valideerimisandmetega on võimalik hinnata mudeli jõudlust ja täpsust õigete otsuste tegemisel. See muutub iteratiivseks protsessiks, kus mudel alguses õpib õppimisandmetest ja seejärel valideeritakse ning viimistletakse valideerimisandmete

abil saadud mudelit. Valideerimiskomplekt ütleb meile, kui hästi mudel õpib ja kohaneb, võimaldades mudeli parameetrite kohandamist ja optimeerimist enne, kui see lõpuks testitakse.

Lõpuks on testimisandmestikust eraldiseisev prooviandmestik, tegemist on mudeli poolt nägemata andmestikuga, et pakkuda mudeli sobivuse lõplikku erapooletut hindamist. Testandmete sisendid on sarnased eelnevatele etappidele, kuid mitte samad andmed. Testandmestik peegeldab reaalse maailma andmeid, mida masinõppe mudel pole varem näinud. Selle peamine eesmärk on pakkuda õiglast ja lõplikku hinnangut selle kohta, kuidas mudel käituks uute andmetega kokku puutudes töökeskkonnas.

Treeningandmed said jagatud autoril suhtega: 80% õppimisandmed, 15% valideerimisandmed ja 5% testandmed.

Kõikide OHZ versioonidega sai autoril koostatud eraldi mudel ja kui tootmisliinil vahetatakse tootetava toote versioon, siis vahetas ka kaameraprogramm ennast automaatselt, aktiveerides selle neurovõrgu mudeli, mis versiooniga OHZ parasjagu tootmises oli.

6.4 Koordinaatteljestikkude konverteerimine

Kaameraprogramm, mis määrab eelnevas peatükis seletatud viisil detaili nurga ja asukoha ning saadab need andmed etteanduri kontrollerprogrammi, mis konverteerib need roboti koordinaatteljestikku. Esiteks lõi autor robotile uue tasandi, tasand robotil tähendab ühte punkti ruumis, mille järgi luuakse uus koordinaatteljestik. Seda nimetatakse tasandiks, kuna selle määramine käib kolme punkti alusel, et võimalikult täpselt määrata tasandi Rx, Ry ja Rz komponendid. Autori olukorras tähendas see seda, et tasand sai määratud kasutades paindlaua enda pinda ning tänu sellele oli tasandi orientatsioon täpselt sama, mis paindlaual. Siis liikudes roboti haaratsiga X-telje suunas positiivselt liigume täpselt mööda paindlaua ühte äärt ja liikudes Y-telje suunas positiivselt liigume samuti täpselt mööda teist äärt, mõlemal korral ei muutu liikumise ajal kõrgus paindlaua ja haaratsi vahel.

Kaamera saadab detailide asukohad pikslites ehk mitu pikslit on detaili keskkoht mõlemast kaadri äärtest. Need väärtused on vaja konverteerida asukohaks varem mainitud roboti tasandil. Esiteks leiame, mitu pikslit kaamera kaadris on 1mm roboti tasandil. Selleks asetas autor detaili paindlaua äärde ja märkis ära, mis on selle detaili piksli-koordinaadid kaadris. Pärast seda liigutas autor detaili 100 mm võrra roboti tasandil positiivse X-telje suunas. Märkis uued piksli-koordinaadid ja siis arvutas, mitu pikslit on 1 mm roboti tasandil

kaamera kaadris. Sama tegevust kordas autor ühe korra veel, aga määras seekord tasandi Y-telje suhte kaamera kaadriga. Lõpptulemuseks sai autor järgmised väärtused:

1mm roboti tasandil = X-teljel:5.79 pikslit, Y-teljel:5.71 pikslit

Kuna kaader sisaldab natuke suuremat ala paindlaua ümber, siis roboti teljestiku 0 punkt oli kaamera piksli-koordinaatides: x:207 y:237 mis tähendab, et lõppvalemis tuleb nihutada positsiooni nende väärtuste võrra. Pannes kokku kõik väärtused saame valemi (*Valem 2*):

$$X' = ((X / 5.79) + 207) / 1000 \qquad (2)$$

$$Y' = ((Y / 5.71) + 237) / 1000$$

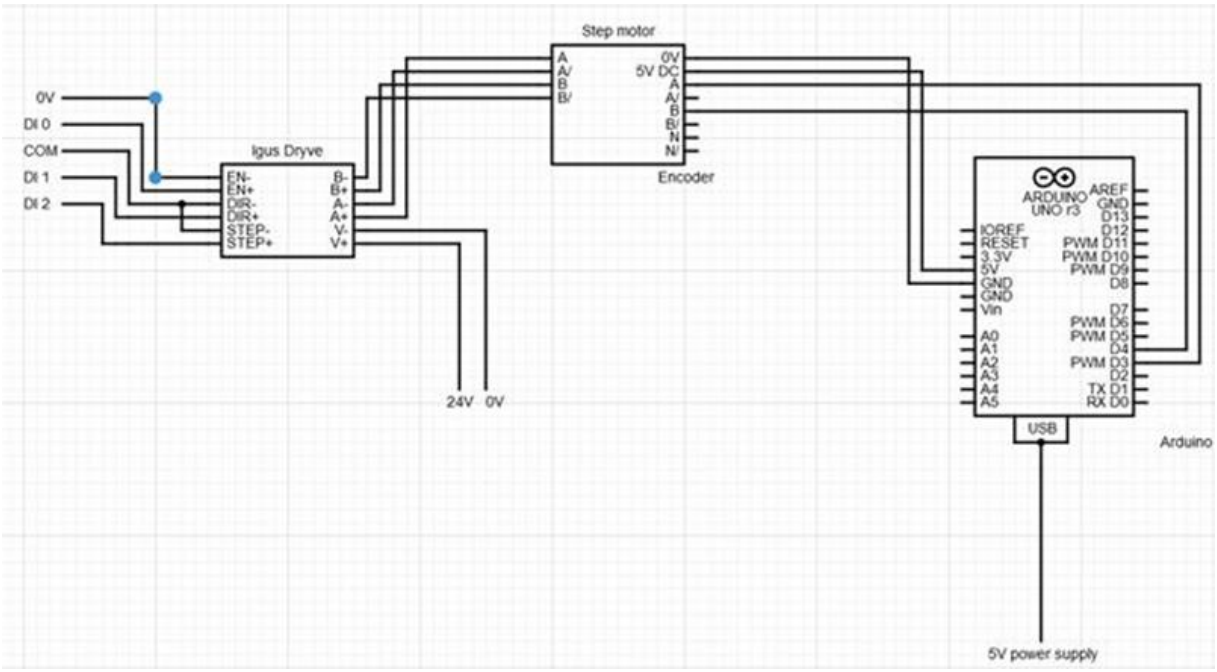
kus X' ja Y' - asukoht roboti koordinaat teljestikus, mm

X ja Y - asukoht kaamera koordinaat teljestikus, pikslid

Valemis 2 on tulemus jagatud 1000-ga, kuna robot kasutab ühikutena ISO standard ühikuid, meetreid. Autor koostas valemi põhjal Phytoni funktsiooni, mis võttis sisendiks kaamera piksli-koordinaadid ja väljundis tagastas vastavad koordinaadid roboti teljestikul. Detaili nurga konverteerimiseks roboti jaoks ei olnud vaja arendada keerulisi valemid, detaili nurk sai konverteeritud programmi poolt radiaanideks ja otse robotile edastatud.

7 LINEAARTELG

Lineaartelg oli projekteeritud positsiooni, et kiirendada tootmistsükli kiirust, telg saab töötada paralleelselt roboti liikumistega ning vabastab roboti vajadusest eemaldada valmistoode rakisest ja asetada see konveierile. Projekti arendusfaasis avastas autor, et enkooder, mis oli mootori külge kinnitatud, töötas hoopis 5V toitepingega, mitte 24V, mis on kõikide teiste loogikaelementide toitepinge ning samuti ka tööstuse standard. Lineaartelg sai hangitud eesti IGUS regionaaltarnija poolt kes seda soovitasid. Tegemist oli täieliku läbikukkumisega nende poolt ning autor pidi kiiresti leidma lahenduse, et sobitata 5V toitepingega enkooderi automaatikapositsiooni. Autor lahendas probleemi kasutades Arduinot. Arduinole saab anda toitepinge läbi USB kaabli, mis lahendas 5V toitepinge probleemi, kuna Arduino sai sellega toita ka enkooderit. Arduino mikrokontrollerid on laialdaselt saadaval ning kiirelt tarnitavad mis lahendas kiiruse probleemi.



Joonis 22. Lineaartelje elektrijoonis

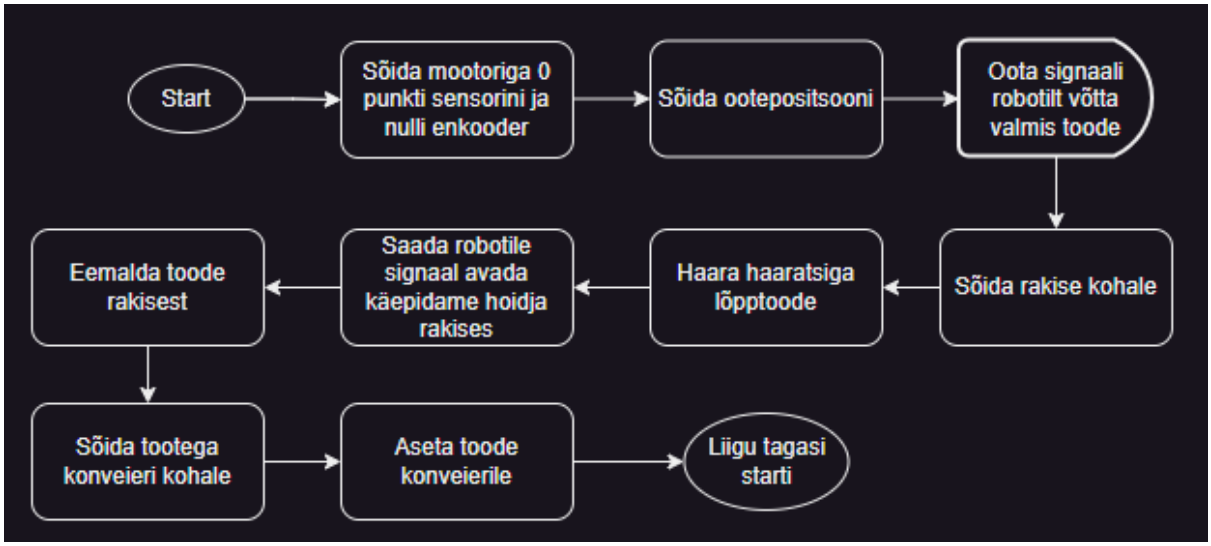
Joonis 22. On näha autori koostatud elektrijoonist lõpplahendusest. Lahendus koosneb kolmest põhilisest komponendist - varem sobitatud IGUS Dryve kontrollerist, stepper mootorist ning sellele kinnitatud enkooderist ja Arduinost. Autori poolt arendatud osa oli Arduino programmeerimine, mis luges impulsi signaale enkooderist ning siis vastavalt suurendas või langetas enda mälus väärtust, mis vastas enkooderi asukohale (Joonis 23).

```
24  void setup() {
25    Ethernet.init(10);
26    Ethernet.begin(mac, ip);
27    delay(1000);
28    mb.server();
29    mb.addHreg(encoderadress0, 0, 1);
30    mb.addHreg(encoderadress1, 0, 1);
31    mb.addCoil(encoderhomingAddress,false,1);
32
33    pinMode(encoderPinA, INPUT_PULLUP);
34    pinMode(encoderPinB, INPUT_PULLUP);
35
36    attachInterrupt(digitalPinToInterrupt(encoderPinA), updateEncoder, RISING);
37
38  }
39
40  void loop() {
41    mb.task(); // Modbus actions, required to work
42    mb.Hreg(encoderadress0,encoder_count & 0xFFFF);
43    mb.Hreg(encoderadress1,encoder_count >> 16);
44
45    if(mb.Coil(encoderhomingAddress)){
46      Home();
47    }
48  }
49
50  void updateEncoder() {
51    // If pin B is low when A trips, then A precedes B
52    // They are reversed for this application
53    if(digitalRead(encoderPinB) == LOW) {
54      encoder_count = encoder_count - 1;
55    }
56    if(digitalRead(encoderPinB) == HIGH) {
57      encoder_count = encoder_count + 1;
58    }
59  }
```

Joonis 23. Kuvatõmmis Arduino programmikoodist

Esiteks kinnitame *interrupt* sündmuse Arduinol olevale pinnile (rida 36). Sündmus on konfigureeritud nii, et kui pinn saab suureneva signaali, siis jooksutab programm funktsiooni „*updateEncoder*". Funktsioon „*updateEncoder*" loeb enkooderist tulevate A ja B signaale ja siis vastavalt kas suurendab või vähendab mälus enkooderi väärtust(rida 50-59).

„*Loop*" funktsiooni jooksutatakse kogu aeg nii kiiresti, kui Arduino riistvara võimaldab. Kui funktsioon jõuab enda lõppu, siis kutsutakse see kohe uuesti välja. Autor programmeeris „*loop*" funktsiooni kirjutama modbus serverisse iga funktsiooni iteratsioonil enkooderi väärtuse. Kusjuures pidi poolitama enkooderi väärtuse kaheks 16 bitiseks väärtuseks. Modbus server võimaldab maksimaalselt ühes registris hoida 16 bitist väärtust aga enkooderi väärtus võib sellest suurem olla. Rida 42 ja 43 on näha, kuidas autor poolitas väärtuse kaheks ja kirjutas selle kahte erinevase registrisse.



Joonis 24. Lineaartelje loogika vooskeem

Lineaartelje loogika oli teiste programmidega võrreldes kõige lineaarsema loogikapuuga. (Joonis 24). Autor programmeeris peaarvutisse RPC serveri, kus iga asukohta liikumine oli eraldi funktsioon. Robotkäsi kutsus neid funktsioone välja enda programmi lõimust.

```
2386    DriveResponse = driveRPC.MoveToPos("Pick")
2387    If DriveResponse≟ False
2389      Wait fixture ≟"released"
2390      Wait: 0.1
2391      DriveResponse = driveRPC.PickupActions()
2392    If DriveResponse≟ False
2394      fixture:="release2"
2395      Wait fixture ≟"released2"
2396      axUID:=fxUID
2397      axfeedfailcount:=FeedFailedCount
2398      FeedFailedCount:=0
2399      Wait: 0.3
2400      DriveResponse = driveRPC.PickupActions2()
2401    If DriveResponse≟ False
2403      DriveResponse = driveRPC.MoveToPos("Place")
2404    If DriveResponse≟ False
2406      Ficture__Free:= True
2407      Pickup_part:= False
2408      removePart:= False
2409      DriveResponse = driveRPC.PutDownActions()
2410    If DriveResponse≟ False
```

Joonis 25. Robotkäe kood lineaartelje juhtimiseks

Joonis 25. On näha UR robotkäe programmi lõimu, mis juhib lineaartelge. Lõimus olev kood jookseb paralleelselt peaprogrammiga. Rida 2386 on näha kuidas robotkäsi annab käsu lineaarteljele liikuda võtmispositsiooni. Rida 2391 ja 2400 kutsub välja võtmiskäsud

ja rida 2409 annab käsu teljel liikuda tagasi. Kõik vahepealsed koodiread kontrollivad robotkäe või rakise staatust ja annavad robotile teada, kui lineaartelg on rakise juurest lahkunud, pärast mida saab seal hakata koostama uut detaili.

KOKKUVÕTE

Selles lõputöös seletatakse autori poolt lahti tarkvara, mille ta programmeeris robotiseeritud tootmisliini positsioonile. Antud positsioon koosneb kahest automaatsest etteandurist ning ühest poolautomaatsest, kasutati UR robotit ja lineaartelge. Automaatikapositsioon suudab toota kuni kolm toodet minutis, vastavalt klientide poolt seatud nõuetele.

Kõigi etteanduritele programmeeritud loogika on põhjalikult selgitatud, esitades igale loogikasüsteemile vooskeemid, mis aitavad keerukaid programme kergemini mõista. Autor kasutas kahes etteanduris masinnägemise elemente, peamiselt detailide asukoha kindlakstegemiseks, kuid ühe etteanduri puhul rakendas ta lisaks ka AI-d, et määrata kindlaks detailide orientatsioon.

Lõputöö keskendub süvitsi tarkvaralisele arendusele, esitledes autorile omaseid meetodeid ja strateegiaid, mida kasutati positsiooni tarkvara väljatöötamisel.

SUMMARY

In this thesis, the author elaborates on how they programmed software for a robotic production line position. This position consisted of two automatic feeders and one semi-automatic, utilizing a UR robot and a linear axis. The automated position can produce up to three products per minute, according to the requirements set by the clients.

The logic programmed for all automatic feeders is thoroughly explained, presenting flowcharts for each logic system to facilitate understanding of complex programs. The author utilized machine vision elements in two feeders, mainly for determining the location of details, but for one feeder, they also employed AI to determine the orientation of the details.

The thesis delves deeply into software development, presenting methods and strategies characteristic of the author used in developing the position software.

VIIDATUD ALLIKAD

[1] P. d. t. Guido van Rossum, „Python Tutorial," 10 Veebruar 2023. [Võrgumaterjal]. Available: https://scicomp.ethz.ch/public/manual/Python/3.9.9/tutorial.pdf. [Kasutatud 03 2024].

[2] T. Zherebetskyy, „Coding Remote Procedure Call (RPC) with Python," Medium, 20 6 2023. [Võrgumaterjal]. Available: https://medium.com/@taraszhere/coding-remote-procedure-call-rpc-with-python-3b14a7d00ac8.

[3] Python Software Foundation, „Python 3.12.2 documentation - Basic XML-RPC servers," Python Software Foundation, [Võrgumaterjal]. Available: https://docs.python.org/3/library/xmlrpc.server.html. [Kasutatud 3 2024].

[4] Zebra Technologies Corp., „Template matching," [Võrgumaterjal]. Available: https://docs.adaptive-vision.com/current/studio/machine_vision_guide/TemplateMatching.html. [Kasutatud 03 2024].

[5] K. D. Foote, „A Brief History of Machine Learning," Dataversity, 3 12 2021. [Võrgumaterjal]. Available: https://www.dataversity.net/a-brief-history-of-machine-learning/.

[6] Zebra Technologies Corp., „Deep learning," [Võrgumaterjal]. Available: https://docs.adaptive-vision.com/current/studio/machine_vision_guide/DeepLearning.html. [Kasutatud 03 2024].

[7] Universal Robots A/S, „User manual - UR10e e-Series - SW 5.12 - English international (en)," 2022. [Võrgumaterjal]. Available: https://www.universal-robots.com/download/manuals-e-seriesur20ur30/user/ur10e/512/user-manual-ur10e-e-series-sw-512-english-international-en/. [Kasutatud 03 2024].

[8] Zebra Technologies Corp., „Aurora Vision Studio," [Võrgumaterjal]. Available: https://www.zebra.com/us/en/products/oem/software/aurora-vision-studio.html. [Kasutatud 03 2024].

[9] Kili Technology, „Training, Validation and Test Sets: How To Split Machine Learning Data,“ Kili Technology, [Võrgumaterjal]. Available: https://kili-technology.com/training-data/training-validation-and-test-sets-how-to-split-machine-learning-data. [Kasutatud 04 2024].

[10] Optimo Robotics OÜ, „Optimo-S2 mini screwdriver,“ [Võrgumaterjal]. Available: https://optimo.ee/product/optimo-s2-mini-screwdriver/. [Kasutatud 04 2024].