TARTU ÜLIKOOL

LOODUS- JA TEHNOLOOGIATEADUSKOND

Füüsika instituut

Hanno Soo

# OPTILINE VÄRVIPURKIDE ASENDIMÄÄRAJA

Bakalaureusetöö (12 EAP)

Juhendaja: Fred Valk, MSc

Kaitsmisele lubatud: .....

Juhendaja: .....

*allkiri, kuupäev*

Tartu 2014

# Sisukord

# SISSEJUHATUS

Iga tänapäevase tootmisettevõtte eesmärgiks on hoida tootmiskulud võimalikult madalad, et tagada toote konkurentsivõimeline hind ja ettevõtte jätkusuutlikkus. Toorme ja elektri hinnad on sageli määratud rahvusvahelise turuhinnaga ja ei ole otseselt ettevõtte poolt mõjutatavad. Inimtööjõu kallinemine on paratamatu eriti riikides, mille keskmine palgatase jääb Euroopa Liidu omale alla. Sellistes oludes saab tootmiskulude vähenemine toimuda üksnes tootmise üldise efektiivsuse tõusu arvelt. Kui uute ja moodsamate tootmisseadmete soetamine pole võimalik, siis jääb üle üksnes olemasolevate tootmis- ja inimressursside optimaalsem kasutamine.

Seoses üldise majanduskriisiga vajasid ettevõtted efektiivsemat tootmist, sest turu nõudlus vähenes, samas tuli ettevõttel teenida kasumit ja püsida konkurentsis. Suurema efektiivsuse saavutamiseks on palju erinevaid võimalusi. Uute tootmistehnoloogiate arendamine on aga tihti väga ressursimahukas ning seetõttu otsitakse teisi võimalusi kulude vähendamiseks.

Üheks võimaluseks tootmiskulusid kontrolli alla hoida on olemasolevate tootmisressursside optimaalsem kasutamine. Samas sunnib palgakulude pidev tõus tootmisettevõtteid otsima ka uusi võimalusi uute tehnoloogiate kasutusele võtuks nii oma toodetes kui ka oma tootmises. Uueks tehnoloogiaks võib kindlasti lugeda erinevad kontaktivabad mõõteseadmed, näiteks kontaktivaba termomeeter. Kontaktivabad mõõteseadmed on palju vastupidavamad kulumisele ja vajavad vähem hooldust ja üleüldist tähelepanu. See aga omakorda vähendab palgakulusid ning tõstab efektiivsust, sest üldist asjaajamist, nagu tootmise ringi planeerimine, varuosade tellimine jms on vähem.

Käesoleva bakalaureusetöö eesmärgiks on projekteerida riistvara ja tarkvara, mis asendaks elektrilisel kontaktil põhineva värvipurgi asendi määramise automaatika optilise asendi määramisega. Elektrilise meetodi puuduseks on asjaolu, et see vajab elektrilist kontakti ja kulub kiiresti. Elektriline kontakt pole aga võimalik läbi purgi etiketi, mis tähendab, et etikett ei saa katta kogu purki. Optilise meetodi eeliseks on asjaolu, et otsene mehaaniline kontakt puudub, mis omakorda tagab süsteemi vähese kulumise. Asendi määramiseks kasutatakse etiketile trükitud spetsiaalset koodi, mida optiliselt loetakse. Samas optilise meetodi puhul pole vahet, kas asendi määramiseks kasutatav kood on paberetiketil või on otse purgile värvitud. Saadud optiline signaal muundatakse seejärel elektriliseks signaaliks, mida

töödeldakse mikrokontrolleriga spetsiaalselt selle ülesande jaoks loodud tarkvaraga. Esmalt saadud signaal puhastatakse võimalikust mürast, kasutades digitaalse signaalitöötluse meetodeid. Seejärel määratakse signaali järgi purgi asend, kasutades digitaalse korrelatsiooni meetodit.

Selline süsteem paigutub manussüsteemide (ingl *embedded systems*) alla. Manussüsteemid on mikrokontrolleril põhinevad rakendused, mis on tavaliselt valmistatud täitma mingit konkreetset ülesannet. Nende kasutajaliides on vähese paindlikkusega või isegi puudub, sest süsteemid võivad olla täielikult autonoomsed.

Töö esimeses osas antakse ülevaade manussüsteemidest ning kontaktivabadest mõõtemeetoditest ja vastavatest seadmetest. Esimeses peatükis lähenetakse tööle teoreetiliselt, et koguda informatsiooni nõutud manussüsteemi projekteerimiseks. Teises peatükis püstitatakse manussüsteemile esitatavad nõuded, projekteeritakse elektroonika ja kirjeldatakse loodud tarkvara. Viimane peatükk tegeleb seadme vastavuse hindamisega püstitatud nõuetele.

# 1. ÜLEVAADE MANUSSÜSTEEMIDEST

Mikroprotsessor leiutati õnneliku juhuse läbi, kui arendati 1970ndatel kalkulaatorite kiibistikele programmeeritavat asendust. Eelnevalt oli kasutusel üksikutest loogikalülitustest kokku pandud lahendus, mis võis koosneda sadadest väratitest. Algne mikroprotsessor sisaldas erinevaid loogikalülitusi ühes kiibis, mida sai programselt valida. Tehnoloogia arenedes suudeti mahutada üha rohkem funktsionaalsust samasse kiipi. Näiteks lisati valmis täissummaator selle ise loogikalülitustest koostamise asemel. See võimaldas toota madala hinnaga kiipi, mis võimaldas täita suure kiibistiku ülesandeid. [4]

Sõna mikroprotsessor mainimine tavaliselt manab esile kujutluse personaalarvutist, millel töötab tekstiredaktor või mõni muu kasutaja tarkvara. Samas on mikroprotsessorid kasutuses ka paljudes muudes seadmetes peale personaalarvuti, enamjaolt mikrokontrolleritena. Näiteks võib tuua pesumasinad, autod või mänguasjad. Mikroprotsessoriteta oleks neid palju keerulisem toota ning ka hinnad oleks vastavalt kõrgemad. [4]

Manussüsteemid ongi mikrokontrolleril põhinevad süsteemid, mis on mõeldud täitma üksikut või väheseid funktsioone. See võimaldab väga täpselt valida, mis on oluline süsteemi tööks, ning luua optimaalne süsteem, mis on vähemate üleliigsete osade tõttu odavam toota. Samas prototüüpide loomise tarbeks luuakse arendusplaate, millesse on sisse ehitatud manussüsteemides tihti kasutatav funktsionaalsus, millest saab programmi abil valida sobiva. See omadus muudab lihtsamaks manussüsteeme muuta ja uuendada, kuna tihtipeale saab asendada vaid tarkvara. [4]

Manussüsteemid võimaldavad väga täpset juhtimist, mis on oluline igale mehaanilisi osi sisaldavale süsteemile. See lubab kasutada tagasisidet ja takistab ning vahel isegi kompenseerib liigset mehaanilist kulumist. Sobivaks näiteks võib tuua mootori juhtsüsteemi. Manussüsteem saab jälgida mootori olekut läbi parameetrite, nagu temperatuur ja pedaalide positsioon ning nende põhjal juhtida mootoris toimuvaid protsesse, nagu kütuse sissepritse jm. Tagasiside võimaldab palju efektiivsemat juhtimist. Lisaks on manussüsteemi võimalik seadistada erinevate prioriteetidega, nagu näiteks kiirendus, kütusekulu, võimsus jne. Mootori kulumisel saab süsteem automaatselt kompenseerida juhtimist või väga suure kulumise korral teavitada juhti tarvilikust hooldusest. [4]

## 1.1. Programmeeritav loogikakontroller

Enne programmeeritavaid loogikakontrollereid enamik tööstuslikku automaatikat oli teostatud keeruliste relee, järjestajate ja tagasiside skeemide abil. Iga erineva osa ja jupi jaoks oli tarvis projekteerida täiesti uus juhtskeem, mis rakendaks teistsugust loogikat. [5]

Digitaalsete arvutite arenguga asendati need keerulised süsteemid üldotstarbeliste arvutitega. Varajased arvutid vajasid palju modifitseerimist tööstuslikus keskkonnas töötamiseks. Neid tuli kaitsta suurema vibratsiooni ja temperatuuri muutuste eest, vajalik oli lisada mitmeid kiiresti ligi pääsetavaid suurt võimsust kannatavaid digitaalseid sisendeid ja väljundeid. Lõpetuseks oli nende kasutajaliides ikka piisavalt keeruline, et tavapärased töölised ei olnud võimelised neid korralikult opereerima ning tuli palgata spetsialiste. [5]

Taoliste probleemide lahendamiseks loodi esimesed programmeeritavad loogikakontrollerid (ingl *programmable logic controller* – PLC). Need on mikroprotsessoril põhinevad arvutisüsteemid, mis tegelevad peamiselt tööstuslike juhtsüsteemidele vajaliku loogika realiseerimisega. PLC tähtsaimad omadused on järgmised:

- Deterministlik loogika – Väga oluliseks tööstusliku juhtimise juures on eeldus, et iga kindla sisendite kombinatsiooni korral tuleb väljund alati sama kiiresti ja sama väärtusega. Ei oleks mõeldav, et tonnide viisi metalli liigutavad masinad määramatult ringi sõidaksid. Seega ei kasuta tööstuslike loogikakontrollerite käsustikus näiteks katkestusi.[6]
- Robustne ehitus – Need kontrollerid on mõeldud töötama tööstuslikus keskkonnas, kus võib olla palju segavaid faktoreid, nagu tolm, niiskus, vibratsioon, suur temperatuuri kõikumine, tugev elektromagnetmüra jm. Selle lahenduseks koosnevad tööstuslikud loogikakontrollerid tihti vähestest tugevatest osadest ning on loodud lihtsa struktuuriga.[6]
- Palju sisendeid ja väljundeid – Kuna tööstuslikud loogikakontrollerid on mõeldud asendama just keerukaid juhtsüsteeme, mis juhivad paljusid erinevaid seadmeid, siis on tarvis, et kõik seadmed saaksid kontrolleriga ühendust. Lisaks peab kontroller arvestama paljude erinevate sisendseadmetega, nagu näiteks lülitid, temperatuuri- või rõhusensorid, keerukad positsiooni määravad sensorid või isegi masinnägemist võimaldavad seadmed. Tihtipeale on loogikakontrollerid loodud modulaarse

süsteemina, kuhu saab vajadusel lisasisendeid või -väljundeid juurde ühendada uut kontrollerit muretsemata.[6]

- Lihtsustatud kujul programmeerimine – tööstuslike loogikakontrollerite kasutajateks on peamiselt mõeldud insenerid, kellel tihti puudub programmeerimise ja arvutitehnika taust. Kuna digitaalsed juhtimissüsteemid on üsna keerulised, siis on nende kontrollerite valmisprogrammeeritud käsustik loodud võimalikult sarnane traditsioonilise nn redeliloogikaga. Enamik tööstuslike loogikakontrollerite kasutusjuhtudest on peamiselt seotud järjestikuste tegevuste ajastamisega.[6]
- Programmeeritavus – Iga süsteemi täpse töö otsustab lõplikult programm, mida saab vastavalt vajadusele teha lihtsam või keerulisem. Kui tootmisliinil peaks tehtama muudatusi saab peaaegu ala edasi kasutada sama seadet, muutes vaid koodi.[6]

Kuigi tööstuslikud loogikakontrollerid on väga heaks lahenduseks suurtele paljude sisendite ja väljunditega süsteemidele, ei leidu nende hulgas sobilikku käesoleva töö lahenduseks. Kuna puuduvad katkestused ja vabalt kasutatav sisemine mälu, siis ei ole võimalik keeruliste mustrite tuvastamine.

# 2. SÜSTEEMI PROJEKTEERIMINE

## 2.1. Süsteemi funktsionaalsed nõuded

Käesoleva projekti eesmärgiks on projekteerida manussüsteem värvipurkide orientatsiooni määramiseks tootmisliinil. Järgnevalt kirjeldatakse süsteemi funktsionaalsusele seatud nõudeid.

- Süsteem peab tuvastama pöörleva purgi peal oleva spetsiaalse ribakoodi ning selle tuvastamisel andma väljundisse ettemääratud pikkusega pulsi, mis juhib liinil olevat automaatikat. Ribakood koosneb neljast mustast kriipsust, mille vahel ja äärtes on valged kriipsud. Ühe kriipsu pikkus on täpselt 3 mm. Tarkvara määrab, millal viimane must kriips lõppeb ja valge kriips algab.
- Koodi tuvastamise ja väljundpulsi andmise vahepealne aeg peab olema muutumatu pikkusega, mida on võimalik seadistada töö käigus.
- Väljundpulsiks on NPN signaal muutumatu pikkusega, mis on seadistatav vahemikus 1-100 ms.
- Süsteem salvestab ning kuvab ekraanil kasutajale tuvastatud toodete hulka. Seda loendurit saab kasutaja nullida.
- Järgmiseid süsteemi parameetreid peab olema võimalik kasutaja poolt seadistada ning need peavad säilima ka peale toite välja lülitamist:
    1. Viide ribakoodi ja väljundpulsi andmise koha vahel;
    2. Ooteaeg enne magamisrežiimi aktiveerimist;
    3. Väljundpulsi pikkus;
    4. Minimaalne toodete vahetumise aeg.
- Kuna süsteem peab töötama vabrikus, siis ei tohi see sõltuda keskkonnatingimustest nagu näiteks valgustatus, vibratsioon jms. Lisaks peaks toitepinge (alalispinge) olema vahemikus 15-35 VDC, eelistatult 24 VDC.

- Kuna purkide suurused on erinevad, peab purgi asendit olema võimalik määrata erinevatelt kaugustelt. Antud juhul vahemikus 4-30 cm. See tähendab, et optilise sensori ja purgi vahekaugus peab jääma vahemikku 4-30 cm.

- Süsteem peab olema piisavalt kiire, et purgi pööramisel ka vastav kood jõutakse maha lugeda.

## 2.2. Riistvara projekteerimine ja kasutatavad komponendid

### 2.2.1. Andur

Kõige olulisemaks süsteemi osaks on andur, mis suudaks eristada heledaid ja tumedaid osi purgil. Sellisteks on kontrastisensorid. Neil on tavaliselt sisseehitatud valgusallikas, mis suunatakse mõõdetavale pinnale ning mille tagasipeegeldunud kiirguse tugevus muundatakse elektriliseks signaaliks. Meile sobiv andur peab sisendile väga kiiresti reageerima. 30 cm ümbermõõduga purgi tavapärane pöörlemiskiirus on ligikaudu 1 pööre sekundis. 3 mm laiuse kriipsu möödumiseks kulub siis: $\frac{3\,mm}{30\,cm/s}=10\,ms$ . Andur peaks lülitama vähemalt kümme korda kiiremini ehk vähemalt 1ms, siis on võimalik eristada ribakoodi stabiilseid lülitusi etiketil olevast tekstist. Kiirema lülitamise korral saab koodi asukoht täpsemini määratud. Täpsuse tagamiseks on oluline ka sensori mõõtmisala ehk tekitatud valgustäpi suurus. Kuna anduri väljund kajastab mõõtmisala keskmist heledust, siis väiksema täpiga andur kujutab reaalsust paremini. Käesolevas lahenduses sobib hästi digitaalse väljundiga andur, kuna uuritav kood koosneb vaid kahest erineva kontrastsusega toonist. Lisaks peab sensor olema võimalikult müra- ja häirekindel.

Selleks valisime anduriks Omron E3Z-LL63-2M optiline sensori. Tegemist on kontrastisensoriga, mis saadab välja laserikiire ning mõõdab tagasi peegeldunud kiirguse intensiivsust. Olenevalt sellest, kui suur on tagasipeegeldunud kiirguse intensiivsus, seatakse väljund kas kõrgesse (toitepinge) või madalasse olekusse (0 VDC). Lülitamiseks kuluv aeg on 0,5 ms (0,0005 s), millele vastab sisendsignaali maksimaalne sagedus 1 kHz. (Ühe täisperioodiga peab lülitama nii alla kui üles st 1 ms kokku.)

- Sensori täpne mudelinumber: E3Z-LL63-2M
- Sensori tööpinge: 12-24 VDC, maks. kuni 26 VDC.

- Sensori poolt tarbitav vool: kuni 30 mA.
- Maksimaalne kiirus: väljundsignaali oleku muutmiseks kulub 0,5 ms.
- Maksimaalne sagedus: 1 kHz. Kiiremat laserkiire oleku muutust väljundis ei kajastata.
- Väljundi tüüp: transistor, NPN.
- Väljundsignaal: Kõrge oleku korral tõmmatakse väljundsignaali klemm nn maha. Sinna saab lasta voolu kuni 100 mA. Madala oleku puhul väljund-signaali klemmi voolu ei lubata. Režiimi saab valida D-L nupust vt allpool.
- Ühenduspistik: Sellel mudelil tuleb välja juhe.
- Ülemine regulaator (min-max): Võimaldab valida, mis vahemikus tagasi peegeldunud kiirt loetakse.
- Alumine regulaator (D-L): Võimaldab valida, kas väljund on kõrges olekus musta (D) või valge (L) sildi nägemisel. Korrektseks tööks peab olema seatud L.
- Roheline tuli (LED): Näitab stabiilsust st kas sensor suudab heledal ja tumedal pinnal vahet teha.
- Kollane tuli (LED): Süttib siis, kui väljundsignaali olek on madal st hele (L).

Lasersensor tuleb ühendada koodilugeja külge M8 pistikuga. Kui lasersensor on ühendamata, siis kuvatakse ekraanil vastav veateade. Oluline on, et roheline tuli alati põleks (ka vilkuda ei tohi) ja kollane tuli põleks siis, kui laseri kiir on heleda pinna peal. Tumeda pinna peal peab roheline tuli ikka põlema aga kollane peab olema kustus. Kui see nii ei ole, siis ei suuda ka tarkvara koodi otsida, sest ta ei saa laserist õiget signaali.

Pilt 1. Foto lasersensorist

### 2.2.2. Kontroller

Lasersensori väljundi analüüsimiseks ja muu funktsionaalsuse tagamiseks on tarvis kontrollerit, millel oleks vähemalt kaks välist katkestust (mida kasutavad lasersensor ja nupp), ligikaudu 25 kB programmimälu ning kasutaja seadete jaoks mitmeid kirjutustsükleid võimaldav mittekustuv mälu. Lisaks on olulisteks punktideks piisav võimsus lasersensori signaalide töötlemiseks ning madal voolutarve. Kõige paremini sobivaks osutus ATmega328P. See on kaheksabitine RISC käsustikuga AVR mikrokontroller, millel on 32 kB Flash mälu (millest 0,5 kB kasutab alglaadur), 2 kB SRAM ja 1 kB EEPROM, mis lubab kuni 100000 kustutamis/kirjutamistsüklit. Kasutada on 14 digitaalset sisendit/väljundit, millest kahe kaudu on võimalik vastu võtta väliseid katkestusi, ning 6 analoogsisendit. Toiteks sobib 6-20 V. Siinkohal võiks ära mainida ka CORTEX-M0 arendusplaadil NXP LPC11U24 MCU. Sellel oleks samuti 32 kB flash mälu ning rohkem arvutusvõimsust 32-bitise arhitektuuri tõttu. Kuid peamisteks puudusteks jääb sisseehitatud EEPROM puudumine ning suurem hind.

### 2.2.3. Nupp

Süsteemi seadete muutmiseks on vaja kasutajalt sisendit saada. Lihtsuse mõttes kasutame vaid ühte nuppu. Kuna seadeid on ühe nupu kohta palju, siis valisime pööratava vajutusfunktsiooniga nupu. See võimaldab pööramisel muuta parameetri väärtust ning vajutamisega liikuda parameetrite vahel. Kõige rohkem muudetav seade (viide) peab olema kõige paremini ligi pääsetav, seega paigutame selle tavaolekus pööramisfunktsiooni alla. Ülejäänud seadeid on tarvis muuta harva, tõenäoliselt vaid süsteemi paigaldamisel.

### 2.2.4. Ekraan

Selleks, et süsteemi tööd oleks võimalik jälgida, ning kasutajale näidata, mis väärtusega on hetkel seadistatavad parameetrid, vajab süsteem indikaatorit. Kuna parameetrid on numbriliste väärtustega, siis ainult vilkuvatest tuledest ei piisaks. Sellepärast vajab süsteem ekraani. Ekraaniks on 20 sümbolit lai ja 4 sümbolit kõrge taustvalgustusega vedelkristallkuvar (ingl *LCD – liquid crystal display*). Valime ekraani, mis kasutab $I^2C$ ühendusprotokolli, kuna ATmega328P mikrokontrolleri programmeerimisel on võimalik kasutada valmis teeki, mis lihtsustab märgatavalt tarkvara arendust.

### 2.2.5. Väljund

Väljundsignaaliks on NPN signaal, ehk lülitamise ajal ühendatakse väljund maaga, muul ajal on väljund ühendamata. Sellise signaali andmiseks kasutame n-kanaliga MOSFET tüüpi transistorit Q1, mis on ühendatud mikrokontrolleri jala 9 külge (vt Lisa 1). Seda jalga saab tarkvaraliselt kas pingestada või mitte. Pingestatud olekus muutub transistor juhtivaks ja ühendab vooluringi väljundis. Pingestamata olekus transistori takistus kasvab ja väljundi vooluring katkestatakse. Väljundtransistori maksimaalseks pingeks on 40 V, lülituskiiruseks < 1 µs. Süsteemi siseselt voolu ei piirata, seega tuleb väljundseadme ühendamisel kontrollida, et kasutatav vool jääks alla 1 A piiri.

Seadmel on väljundiks kaks DIN-5 pesa. Need said valitud nende robustse ehituse ning laialdase kasutuse tõttu tööstuslike seadmete ühendamisel. Süsteemil on kaks väljundpesa, et suurendada ühilduvust rohkemate erinevate seadmetega. Ülemisel antakse signaal välja ekraani poolsest äärmisest klemmist. Alumisel pesal on vastupidi. Teised (3 keskmist) klemmid on ühendamata.

### 2.2.6. Toide

Kogu süsteemi toitepinge (alalispinge) peaks olema vahemikus 15−35 VDC. Soovitavalt 24 VDC. Kuna aga nii valitud andur kui ka mikrokontroller töötavad madalama pingega, siis on vajalik toitepinge sobivaks muundada. Selle tarbeks on kasutusel LM2596S kiibil põhinev impulss-toiteplokk. See võimaldab muundada sisendpinget 4−35 VDC väljundpingeks 1,25−30 VDC. Sensori vähimaks toitepingeks on 12 VDC. Seega seame väljundpingeks 12,5 VDC. Mikrokontrolleri arendusplaadil on sisseehitatud muundi, mis võimaldab seda toita kõrgema pingega kui 5 VDC, kuid see võib üle kuumeneda. Seega lisame toiteploki väljundi ja kontrolleri sisendi vahele 33 Ω takisti R3 (vt Lisa 1). Nii valitud nupp kui ekraan kasutavad toitepingeks 5 VDC, mille saame arendusplaadilt.

Süsteemi voolutarve tuleb < 100 mA, kusjuures 90% voolust tarbivad lasersensor (30 mA) ja ekraan (40 mA). Toitepistiku ja -pesa mõõdud on 2.1 mm x 5.5 mm. (+) klemm tuleb ühendada keskele ja (-) klemm äärde (vt Lisa 1). Kuna (-) klemm on ühendatud seadme sees maaga (st seadme korpusega) kokku, tuleb kontrollida, et ülejäänud süsteem, kuhu koodilugejat paigaldatakse, toetaks sellist lahendust, muidu võib tekkida lühis.

## 2.3. Sensori sidestamine ja koodi leidmine

Optiline sensor on ühendatud ATMega328P mikrokontrolleri jala 3 külge (vt Lisa 1), mis vastab välisele katkestusele 1. Selleks, et mikrokontrolleri jala olek vastaks alati sensori enda väljundi olekuga tuleb sensor õigesti sidestada. Kuna sensori väljundiks on bipolaarse transistori kollektor (tööstuses nimetatakse seda NPN väljundiks), siis tuleb meil sensori nn väljundsignaali jalg tõmmata kõrgesse olekusse. Kui sensor lülitab, siis tõmbab sensor selle jala omakorda madalasse olekusse (vt Lisa 1). Kuna sensorisse maksimaalselt lubatav vool on 100 mA, siis peame seda takisti valikul (skeemil R4) arvestama. Üks võimalus mikrokontrolleri jalg kõrgesse olekusse saada oleks kasutada mikrokontrolleri endas sees olevaid takisteid, mis lubavad tarkvaraliselt suvalist jalga kõrgesse olekusse panna (ingl *pullup*). Nende takistite takistused on aga suured, suurusjärgus 10k oomi. See tähendab, väga väike energia suudab juba olekut muuta, mis pole aga antud lahenduse jaoks sobiv, sest meil on vaja tagada ka süsteemi piisav mürakindlus. R4 takistuseks valime 1000 oomi. Sellisel juhul on sensorist läbi voolava voolu maksimaalseks väärtuseks 5 V/1000 Ω = 5 mA. Oleku muutmiseks on siis energiat vaja: 5 V · 5 mA = 25 mW. Jääb vaid loota, et süsteemi töökeskkonnas pole müral piisavalt energiat, et mikrokontrolleri jala olekut nii muuta, et see ei vasta tegelikule signaalile. Vajadusel saame alati R4 takistust vähendada.

Kuna sensori väljundsignaali oleku muutmiseks kulub 0,5 ms [8], siis järelikult kiiremad oleku muutused ei saa vastata tegelikule signaalile ja saavad seega olla seotud ainult müraga. Kiirete mürapulsside eemaldamiseks kasutame kondensaatorit C5 (vt Lisa 1). Nüüd moodustavad takisti R4 ja C5 RC-ahela ajateguriga $\tau$. Ajateguri saame leida valemist:

$$\tau = R \cdot C \tag{1}$$

Kuna palju kiiremaid muutuseid kui 0,5 ms me ei taha, siis valime $\tau$ = 0,1 ms. See on selle pärast, et ajategur näitab aega, mille jooksul pinge kondensaatoril C muutub e $\approx$ 2.71 korda, kui tema laadumine/tühjenemine toimub läbi takisti R. Meie aga tahame, et see RC-ahel ei segaks sensori enda lülitamist st valime ajateguri paar korda väiksema, kui on sensori enda lülitamise aeg. Asetades suurused valemisse (1), saame C5 väärtuseks: 100 nF. Just sellise väärtusega kondensaatorit skeemil kasutamegi (vt Lisa 1). Kondensaatorina kasutame tavalist plaatkondensaatorit, sest selle laadimine ja tühjenemine on kiire võrreldes elektrolüütkondensaatoritega.

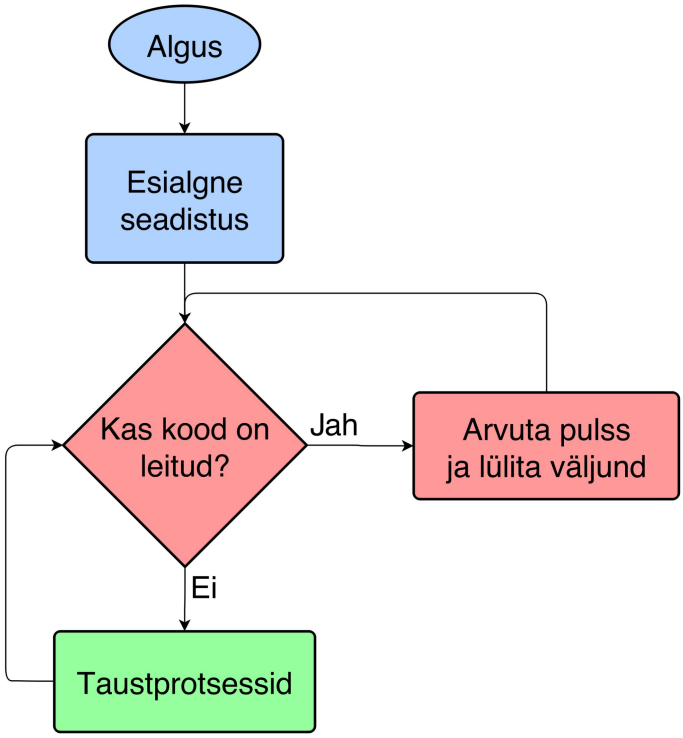
Kui nüüd sensor olekut muudab (tume pind vaheldub heledaga või vastupidi), siis peab muutuma ka mikrokontrolleri jala 3 olek. Selleks, et seda tarkvaraliselt registreerida, teeme tarkvaras järgmised seadistused:

```
pinMode (3, INPUT);
attachInterrupt (1, katkestus, CHANGE);
```

Nüüd töötab jalg 3 sisendina ja jala 3 oleku muutumisel käivitub alamprogramm katkestus(); mida kirjeldame purgi asendi määramise algoritmi juures.

## 2.4. Tarkvara ülesehitus

Tarkvara on jaotatud kolmeks osaks nagu kirjeldab joonis 1. Kõige suuremaks prioriteediks on koodi otsimine ja väljundpulsi tegemine, mis on joonisel kujutatud punasena. Madalama prioriteediga täidetakse taustprotesside ülesandeid (joonisel roheline). Kolmanda eraldi osana loeme seadme käivitamisel tehtavat seadistust, mille ajal koodi ei otsita.




Joonis 1. Tarkvara plokkskeem

## 2.5. Purgi asendi määramise algoritm

Purgi asendi määramise algoritm saab alguse katkestusel 1 täidetavast alamprogrammist katkestus(); See alamprogramm näeb välja järgmine:

```
void katkestus() {
                    katkestusi++;
                    if (!LYLITA && !SLEEP) {
                                                unsigned long lugem = annaAeg();
                                                if(lugem < 30    ) { return; }
                                                if(lugem > 30000) { return; }
                                                nihutaPuhvrit();
                                                lugemid[0] = lugem;
                                                dekodeeri();
                                                }
                    }
```

Kood 1.1. Välise katkestusega 1 seotud alamprogramm.

Lugemi saamiseks on kasutusel alamprogramm annaAeg(); mis võrdleb eelmise lugemi võtmise aega hetkel kontrolleri mikrosekundite registris oleva väärtusega ning väljastab nende vahe 100 korda väiksema väärtuse. Seega lugem tähistab jala 3 kahe muutuse vahelist aega 0,1 ms täpsusega. Funktsioon annaAeg(); on selline:

```
unsigned long annaAeg () {
                          unsigned long praegu  = micros();
                          enne1                 = millis();
                          unsigned long vahe    = (praegu - enne10)/100;
                          enne10 = praegu; return vahe;
                          }
```

Kood 1.2. Välise katkestusega 1 seotud alamprogramm, mis tegeleb aja lugemisega.

Katkestuse alamprogrammist näeme, et kui sensori oleku muutuste vahe on väiksem kui 3 ms või rohkem kui 3 s, siis me ei pane seda pulssi lugemite vektorisse ja ei hakka üldse koodi otsima, st dekodeeri() ; funktsioonini me ei jõua. See on selleks, et välistada väga lühikesed ja väga pikad pulsid, mis ei saa vastata kuidagi tegelikule signaalile, sest purki lihtsalt ei pöörata nii aeglaselt ega kiiresti. Sellised pulsid võivad aga vastata tekstile etiketil, mida meil vaja ei ole.

```
void dekodeeri() {
      kesk  =  mean(lugemid, filtriPikkus);
      halve =  stdv(lugemid, filtriPikkus);
      if((kesk < 30) || (kesk > 30000)) {  return; }
      suhe  =  ((halve*10.00) / kesk);
      if (digitalRead(3)==0) {
         if((suhe < (TUNDLIKKUS))) {
              if (millis() < (enne_p + purkideVahe)) return;
              enne_p = millis();
              LYLITA = true;
              velo   = (3.00 / kesk);
              VIIDE_A= (VIIDE_P/velo)*10;
              katkestusi=0;
              return;}
         }
    }
```

Kood 1.3. Välise katkestusega 1 seotud alamprogramm, mis tegeleb otsitava koodi tuvastamisega.

Süsteem otsib 7 ühepikkust pulssi, kusjuures viimane pulss peab olema must. Ribakood koosneb neljast mustast kriipsust, mille vahel ja äärtes on valged kriipsud. Ühe kriipsu pikkus on täpselt 3 mm. Kui nüüd lasta laseri kiirel joosta üle koodi, siis 7 ühesugust pulssi (viimane must) saadakse täis täpselt sellel hetkel, kui laseri kiir läks viimase musta kriipsu pealt üle viimasele valgele kriipsule. Siis antakse välja väljundpulss (kui viide on 0).

Alati kui sensori olek muutub, käivitatakse viimase musta kriipsu otsimiseks alamprogramm katkestus(); Kui uus leitav pulss ei ole liiga pikk ega lühike, siis käivitub ka alamprogramm dekodeeri(); Esimese asjana leitakse kõikide massiivis lugemid[7] hoitavate pulsside keskväärtus vastavalt valemile:

$$kesk = \frac{1}{6}\sum_{i=0}^{6} x_i \tag{2}$$

Siin $x_i$ on lugemite vektori elemendid. Sarnaselt leiame ka hajuvuse:

$$halve = \sum_{i=0}^{6} \left( x_i - kesk \right) \tag{3}$$

Siin me ei kasuta hälbe hindamiseks standardhälvet sest arvude ruutu võtmine ja jagamine nõuavad palju arvutusressurssi ning meil on hiljem lihtsalt vaja tulemust võrrelda mingi konstandiga. Samuti leiame hälve ja keskväärtuse suhte:

$$suhe = \frac{halve}{kesk} \tag{4}$$

Kui see väärtus läheneb nullile, siis see tähendab, et kõik viimased 7 pulssi on täpselt ühepikkused. Näiteks kui x = (100, 100, 100, 100, 100, 100, 100) · 0,1ms. Siis kesk = 100 ja halve = 0 st suhe = 0. Kui mõni pulss ei ole teistega ühepikkune, siis kohe suhte väärtus tõuseb nt x = (100, 574, 100, 100, 100, 100, 100) · 0,1ms. Siis kesk = 167,7 ja halve = 812,6 st suhe = 48,45.

Kood leitakse juhul kui kõik pulsid on ühepikkused seega peab suhe koodi leidmise hetkel olema väike. Kuna selliseid kohti, kus vastav tingimus on täidetud, on mitu, siis on meil vaja veel lisatingimusi, et koodi asukoha määramine oleks ühene.

Probleemiks on asjaolu, et koodi leidmise koht sõltub sellest, milline on ribakoodi ümbritseva ala värvus. Kui see on valge, siis esimest valget pulssi ei loeta, kui see on must, siis loetakse ka esimene valge pulss, sest tema mõlemas ääres on tume pind. Seega hakkab koodi leidmise koht (7 ühepikkust pulssi) sõltuma sellest, milline on etiketi taustavärvus.

Selle probleemi lahendamiseks nõuame, et viimane pind oleks valge st viimane pulss vastas mustale pinnale. Nüüd ei ole enam vahet, milline on tausta värvus, sest olenemata sellest, kas esimene pulss leiti või mitte, ootame me ikkagi ära viimase valge ala.

Kuna tingimus, et kõik pulsid on täpselt ühepikkused pole reaalsuses kunagi täidetud, sest purki pöörav automaatika pole absoluutselt täpne, siis loeme koodiks koha, millal leitud suhe jääb alla teatud piiri, milleks on tundlikkus. Tundlikkust saab kasutaja nuppu keerates sättida täisarvulisteks väärtusteks 0 – 10. Kõige rangemaks väärtuseks on null, mille korral kood leitakse vaid perfektsetes tingimustes, ja kõige leebemaks 10, mille korral on võimalik koodi simuleerida kätt laseri ees liigutades. Näiteks kui tundlikkus = 3 ja x = (254, 230, 247, 252, 227, 241, 256) 0,1ms, millelt kesk = 24,5 ja halve = 67,1 st suhe = 2,753, siis loetakse see koodiks. Näide vektorist, mis on läbinud 6 pulssi ribakoodist oleks x = (80, 75, 77, 75, 73, 73, 40) 0,1ms. Siis kesk = 70,4 ja halve = 60,9 st suhe = 8,641.

Lisaks saab nupust valida viidet selle hetke vahel, millal viimane must kriips lõppes ja millal väljundpulss välja antakse. Selle viite abil saab valida purgi asendit, millal nt lisaetikett

purgile trükitakse. Purgi pööramise kiirus ja seega ka pulsside pikkus pole seejuures oluline. Tähtis on vaid see, et pulsid oleksid ühepikkused.

Kasutaja poolt seatud viide on esitatud pikkusena, kuid meil on tarvis teada just aega, millal väljundpulss anda. Selle lahenduseks arvutame purgi pöörlemise joonkiiruse. Kui me oleme juba veendunud, et möödus just viimane must kriips, siis iga lugem tähistab kui kaua kulus aega, et mööduks üks kriips. Lisaks teame, et iga kriipsu laius on 3 mm. Sellest arvutame kiiruse valemiga:

$$kiirus = \frac{nihe}{aeg} \tag{5}$$

Siin nihkeks on 3 mm ja ajaks võtame pulsside keskmistatud väärtused, mis on juba arvutatud valemid (2) abil. Tulemuse salvestame muutujasse velo. Rakendame kiiruse valemit teist korda, et sisestatud viite (nihe) ja muutujas velo oleva väärtuse (kiirus) abil leida viiteaeg.

## 2.6. Väljundpulsile vastava ajahetke leidmine

Tarkvaraliselt on väljundi juhtimine lihtne ja lühike, kuid kõige kõrgema prioriteediga protsess. Esmalt kontrollitakse, kas viiteaeg on juba enne meetodi algust mööda läinud. Sellisel juhul teavitatakse kasutajat märkmega ekraani parempoolses ülemises nurgas (ERR1) ning väljutakse meetodist väljundpulssi andmata. Vastasel juhul ootab süsteem tühjas tsüklis kuni viiteaeg on läbi ning vahetatakse väljundi seisundit. Seda seisundit hoitakse muutujas `PULSIPIKKUS` määratud aja, mille järel toimub uuesti vahetus, et tekitada vajalik pulss. Väljundit järgib ekraani taustvalgustus, et kasutajal oleks võimalik ka kaugemalt jälgida pulsi toimumist. Lisaks suurendatakse selles meetodis ka toodete loendureid, kuna väljundpulsi andmine tähistab kindlalt ühe koodi leidmist.

## 2.7. Taustaprotsessid

Seni kuni koodi leitud pole ja viimasest katkestusest on möödunud vähemalt 50 ms, täidetakse vähem ajakriitilisi taustprotsesse. Täitmisele tulevad need järjestikku ning viimase lõpetamisel võetakse käsile jälle esimene.

### 2.7.1. Ajastuse kontroll

Aja mõõtmiseks kasutatakse peamiselt mikrokontrolleris olevate taimerite väärtuseid lugevaid meetodeid millis(); ja micros();. Need väljastavad vastavalt mitu milli- või mikrosekundit on

möödunud programmi käivitamisest. Iga teatud perioodi järel jõuavad need taimerid ületäitumiseni ning alustavad loendamist uuesti nullist. Kasutusel on 32-bitised täisarvud, seega millisekundite taimer saab täis ligikaudu iga $2^{32}$ ms ≈ 50 päeva järel ning mikrosekundite taimer iga $2^{32}$ µs ≈ 70 minuti järel. Seetõttu tuleb tarkvaras kasutusel olevaid ajaga seotud muutujaid ka vajadusel parandada. Iga muutuja kohta kontrollitakse, kas selles talletatud väärtus on suurem kui hetkel taimeris. Sellisel juhul seatakse muutuja praeguse ajaga võrdseks.

Lisaks aegade parandamisele teostatakse selles osas ka magamisseisundisse mineku ja sellest ärkamise kontroll. Magamisseisundis lülitatakse välja ekraan ning kontroller läheb vähese voolutarbega olekusse. Selles olekus tavapäraseid protsesse ei täideta. Olekust väljutakse kas nupust või sensorist tulnud signaali peale.

### 2.7.2. Nupp

Pööratav ja vajutatav nupp on ainukeseks kasutaja sisendi saamise võimaluseks. Kuna kasutusel on elektrilisel kontaktil põhinev riistvara, siis on oluline arvestada kontaktil tekkiva võimaliku põrkumisega. Selle kompenseerimiseks on kasutusel Bounce teek [9]. Teegi kasutamiseks luuakse objekt, mille parameetriteks on digitaalne sisend ning näitude võtmise vaheline aeg millisekundites. Hiljem saab selle objekti abil teada, kas toimus üleminek seisundite vahel või mitte. Taolised objektid on loodud nii pööramis- kui ka vajutamisfunktsiooniga seotud sisenditega.

Programmi tavalise töö käigus nupu keeramine muudab viitepikkust koodi leidmise ja väljundpulsi tekitamise vahel, edaspidi viidet. Kuna seda saab muuta 300 mm ulatuses sammuga 0,1 mm, siis kiirema keeramise korral, kui nupust tulevate signaalide vahe on väike, on muutus iga signaali kohta suurem, kuni 5 mm. Lisaks viitele saab nupu keeramisega muuta tundlikkust, mis tähistab kui sarnased peavad loetud pulsid olema, et tuvastada ribakood. Seda saab muuta nullist kümneni täisarvu kaupa. Tundlikkuse muutmiseks tuleb nuppu keeramise ajal vajutatud olekus hoida. Mõlemad muutused kuvatakse jooksvalt ka ekraanil.

Nupuvajutusel on veel kolm funktsiooni. Tavaline vajutus seab toodete seeria loenduri (loendur, mille väärtus ei püsi üle voolukatkestuse) tagasi nulliks. Kui nuppu peale vajutust veel kaks sekundit all hoida, siis lülitatakse ekraani taustvalgustus välja. Selle saab sama moodi tagasi sisse lülitada. Kolmandaks, kui nuppu all hoida kolm sekundit, teeb süsteem taaskäivituse.

Lisaks saab kindlate viite ja tundlikkuse kombinatsioonidega aktiveerida menüüoleku ja testoleku. Menüü saab aktiveerida kui keerata nii viide kui tundlikkus mõlemad nullideks. Sellisel juhul tuleb ekraanile nimekiri erinevatest muudetavatest parameetritest, millest esimene on valitud ja seda saab nupu kerimisega muuta. Nupuvajutusega valitakse järgmine. Viimase parameetri juures nupuvajutuse korral lahkub programm menüüolekust tagasi tavakäitumise juurde, talletades muudetud parameetrid mikrokontrolleri EEPROM mälus, mille kasutust on detailselt kirjeldatud allpool. Menüüs on muudetavad järgmised neli parameetrit:

- Magamisaeg – 1–100 sammuga 1 minut. See on aeg, mis peab olema möödunud viimasest sensori katkestusest ja nupu muutusest, et süsteem siseneks magamisolekusse.
- Pulsi pikkus – 1–100 sammuga 1 millisekund. Nii kaua hoitakse väljundit muudetud seisundis, kui tuleb anda väljundpulss.
- Pulss – kaks väärtust, kas kõrge või madal. See tähistab, kas väljund on pulssi andes õhus või maandatud.
- Purkide vahe – 0,1–10 sammuga 0,1 sekundit. See tähistab mis aeg peab olema möödunud viimasest koodi leidmisest, et uut koodi oleks võimalik leida. See on vajalik, kuna purki pöörav automaatika ei lõpeta kohe kui pulss saadud on ning purk võib süsteemi ees veel mõned täispöörded teha, lugedes toodete loendurisse üleliigseid tooteid ja anda üleliigseid väljundpulsse. Selle muutuja kaudu saab kasutaja seadistada aja, mille jooksul loetud toode peaks kindlasti süsteemi juurest edasi liikunud olema.

Testoleku aktiveerimiseks tuleb viiteks keerata 1,2 mm ja tundlikkuseks 1. Selles olekus iga nupu vajutus aktiveerib `LYLITA` muutuja ja kohe arvutatakse väljundpulss. Selle abil on võimalik testida väljundahelasse ühendatud seadmeid ilma reaalset koodi leidmata.

### 2.7.3. Ekraani juhtimine

Suhtlus ekraaniga käib kasutades $I^2C$ protokolli. Põhitasemel kirjutamine on realiseeritud Arduino teegis LiquidCrystal_I2C, millest on pärit meetodid ekraanil positsiooni valimiseks ja sümbolite kirjutamiseks [10]. Selleks et vähendada koormust $I^2C$ siini peal on loodud virtuaalne ekraan, mida hoitakse mikrokontrolleri mälus. Selleks on kaks 4 realist 20 sümbolit pikka massiivi. Ühes on hoiul viimane ekraani seis ning teise saab mujal koodis muudatusi

teha. Siis kui jõuab kätte ekraaniga seotud taustprotsess, võrreldakse neid omavahel ning siinile kirjutatakse vaid need sümbolid, mis on muutunud. Kirjutamisega paralleelselt uuendatakse ka ekraani seisu kujutav massiiv. Liiga tihe ekraani uuendamine muudab teksti halvasti loetavaks, seega on selles protsessis kaitse, et ekraani uuendatakse vaid iga 100 ms möödumisel. Kasutaja jaoks märgatavat viidet ei teki ja ekraan paistab stabiilsem.

### 2.7.4. EEPROM mälu juhtimine

Mälust lugemise ja kirjutamise jaoks on üks lisa päisefail, mis lubab üksikute baitide asemel mälust lugeda või sinna kirjutada suvalisi objekte. Selleks arvutatakse vastava objekti suurus baitides ning vastav hulk baite loetakse järjestikku muutujatesse või kirjutatakse mällu. Väga oluline on siinkohas kirjutada ja lugeda täpselt samu objekte, muidu on tulemus määramatu.

Kuna EERPOM mälul on piiratud kustutamis/kirjutamistsüklite arv, siis tuleb hoiduda liigsest mällu kirjutamisest. Selle vältimiseks kasutatakse sarnast süsteemi ekraani värskendamisega. Mitte tihemini kui iga minuti järel võrreldakse salvestatud muutujaid töömälus olevate muutujatega ning vajadusel uuendatakse. Eeldades, et süsteem leiab tooteid 40 tundi nädalas, on mälu kindlasti töökorras ligikaudu 10 kuud. Lugemise jaoks taolisi piiranguid pole. Lisaks iga minutisele kontrollile kirjutatakse seaded mällu menüüst väljudes ja magamisrežiimi aktiveerimisel.

## 2.8. Süsteemi paigaldamine ja seadistamine

Koodilugeja õige seadistamine on väga oluline, sest vastasel korral süsteem lihtsalt ei tööta stabiilselt. Sensori alumine regulaator (D-L) peab olema seatud L. Sensor tuleks paigaldada purgist umbes 30 cm kaugusele, purgi suhtes püsti. Õigel kaugusel on kiir kitsas kriips. Seega saab õiget kaugust määrata kui vaadata, millisel kaugusel laseri kiir just terava püstise kriipsu moodustab. Laseri kiir tuleks suunata koodile sellise nurga all, et tagasipeegeldunud kiir ei satuks otse sensorisse, vastasel korral muutub sensor ebastabiilseks. Kõige parem on, kui peegeldunud kiir satuks mingi musta mati või hajutava pinna peale. Selleks tuleks kiir suunata koodile veidi ülevalt poolt, et siis peegeldunud kiir peegelduks kuhugi alla ära. Hea kui peegeldunud kiir ei satuks mõne metalse pinna peale, kust ta võiks jälle sensorisse tagasi peegelduda. Tuleb ka vaadata, et mõne teise sensori kiir või peegeldus ei satuks sensori vaatevälja. Kui sensor on paigaldatud õigesti, siis purgi pööramisel roheline tuli sensoril kogu aeg põleb, mitte ei vilgu. Heleda pinna (valge kriips) peal peab süttima kollane tuli ja musta

pinna peal peab see kustuma. Sensorit ei tohi paigaldada risti purgiga. Ta peab olema samas asendis nagu purk, sest siis on mõõtmine kõige täpsem. Laseri kiire kriips peab olema sama pidi nagu koodi kriipsudki.

Kui sensor on paigas, siis tuleks purk pöörlema panna ja esialgu tundlikkus võimalikult suur valida. Kui kood igal pöördel leitakse (ekraani taustvalgustus vilgub), siis kõik toimib. Nüüd tuleks sensorit paremini paigutades tundlikkust vähendada. Mida paremini on sensor purgi suhtes paigutatud, seda kergemini st madalama tundlikkusega, kood leitakse. Tundlikkus peab olema võimalikult väike aga nii suur, et kood iga purgi pöördega ikka leitakse.

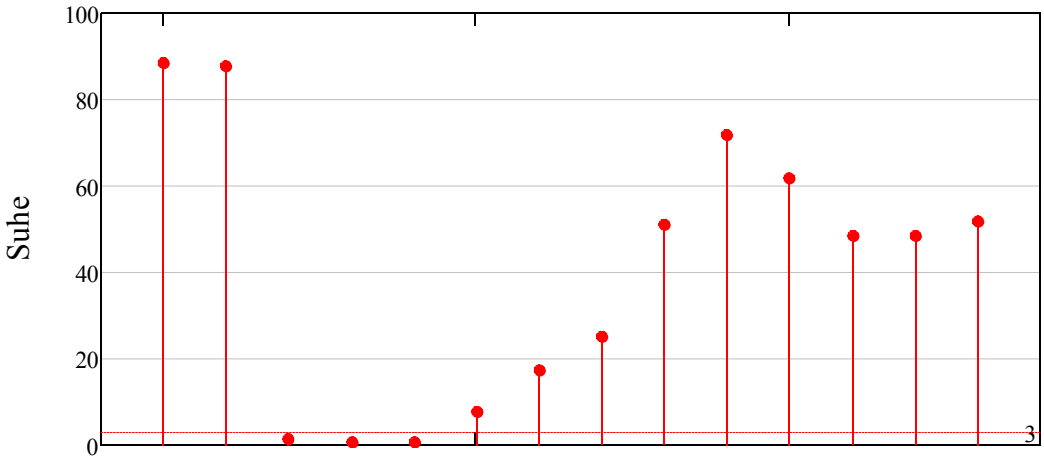
# 3. TULEMUSED JA ANALÜÜS

Selles peatükis analüüsime loodud süsteemi vastavust nõuetele ning erinevaid veaohtlikke olukordi.

Testimise eesmärgil lisasime töötava seadme tarkvarasse võimaluse väljastada järjestikused saadud lugemid ning kasutasime päris tootmises leiduvat purki. Nendest lugemitest võtsime 20 järjestikust elementi nii, et selle hulga keskele jääks ka üks koodi tuvastamise koht. Lugemid on järgmised:

(58, 1459, 77, 73, 71, 73, 71, 71, 73, 72, 71, 41, 145, 140, 557, 1303, 814, 906, 39, 47)

Sellest hulgast saame 13 erinevat seisu, mida süsteem hindama hakkab, näiteks esimeseks on (58, 1459, 77, 73, 71, 73, 71). Igale grupile rakendasime dekodeeri(); meetodis kirjeldatud arvutusi, millest meid huvitab leitud suhe. Väärtused on toodud välja järgmisel joonisel:




Joonis 2. Suhte väärtus erinevate lugemi seisude korral

Siin y-teljel on massiivi lugemid[7] elementide hälbe ja keskväärtuse suhe vastavalt ülal kirjeldatud valemile (4). Lisaks on märgitud tundlikkus, mille väärtuseks on seatud 3. Joonise teine pool kajastab tavapäraseid väärtuseid, mida süsteem saab, kui sensor liigub üle purgi etiketi. Elementide suure erinevuse tõttu on suhe palju üle sobivate tundlikkuste väärtuste ning koodi ei leita. Esimese poole peal aga leidub kolm gruppi, mis sobivad koodiks. Nendeks on kolmas, neljas ja viies grupp. Kuuenda grupi puhul tõuseb suhe juba üle 7 ning õige süsteemi installatsiooni ning seadistuse korral jääb ka see välja. Kuna testimisel oli kasutusel purk, millel ribakood oli asetatud mustale taustale, siis sobibki kolm erinevat gruppi. Seda

selle tõttu, et üleminekud tausta ja äärmiste valgete kriipsude vahel võetakse ka lugemite hulka. Selles olukorras leiab süsteem ikka õige koodi, lisatingimuse tõttu, et viimane üleminek oleks mustalt valgele. Seega eeldades, et kolmas grupp algas üleminekuga tumedalt taustalt esimesele valgele kriipsule, asub laser sellel hetkel viimasel mustal kriipsul ning väljundpulssi ei anna. Kuna seal tegelikult oli kood ja neljas grupp lõpetab valgel kriipsul, siis leiamegi koodi. Tööolukorras järgmisi lugemeid ei leita, kuni väljundpulss on antud ja uus toode on süsteemi ette jõudnud.

Koodi leidmise juures tegime tähelepaneku, et üksiku kriipsu pikkus ja seega ka purgi pöörlemise kiirus ei mõjuta koodi tuvastamise tulemust. Siiski leiduvad riistvaralised piirangud. Seega leiame suurima joonkiiruse, mille korral süsteem veel on suuteline koodi ära tundma. Kuna sensori väljundsignaali oleku muutmiseks kulub 0,5 ms [8], siis tuleb vähimaks lugemiks 5. Järelikult $velo_{max}$ = 3,00/0,5 = 6 m/s. Kuna aga reaalsed lugemid on piiratud väärtusele 30, siis $velo_{max}$ = 3,00/3,0 = 1 m/s. Seega 30 cm ümbermõõduga purk võib maksimaalselt teha 1/0,3 = 3 1/3 pööret sekundis.

Siit järeldub, et õige seadistuse ja kasutuse korral suudab süsteem alati koodi üheselt tuvastada ning valesignaale ei teki.

# KOKKUVÕTE

Käesoleva bakalaureusetöö eesmärgiks oli disainida kasutatav riistvara ning juhttarkvara värvipurkide optilise asendimääramise süsteemile. Süsteem oli mõeldud asendama kasutusel olevat elektrilisel kontaktil põhinevat süsteemi. Esiteks tuli tutvuda manussüsteemide disaini põhimõtete ja digitaalsete arvutisüsteemide kasutamisega tööstuslikus keskkonnas. Saadud teadmisi kasutades, sai koostatud funktsionaalsed ja tehnilised nõuded loodavale süsteemile.

Järgmise sammuna sai valitud ja põhjalikult kirjeldatud olulisemad osad kasutusele võetavast riistvarast. Sensoriks sai Omroni optiline kontrastisensor kiire lülitamisaja ning mikrokontrolleriks ATmega328P sobiva hulga sisendite, sisseehitatud EEPROM mälu ning mõistliku hinna tõttu. Lisaks tuli leida lahendused tööstuses kasutatava pinge juhtimiseks väljundis, erineva toitepingega seadmete (sensor ja mikrokontroller) koos kasutamiseks ning efektiivseks suhtluseks kasutajaga.

Tarkvara disainimisel tuli erilist tähelepanu pöörata sensorist andmete vastuvõtmisele ja koodi leidmise algoritmile. Koodi tuvastamiseks kõigepealt salvestasime viimased seitse sisendpulssi. Võrreldes nende hälvet etteantud väärtusega saime teha otsuse. Peale koodi leidmist arvutasime väljundpulsi andmiseks täpse ajahetke ning sooritasime väljundi lülituse. Lisaks oli tarvis luua hulk taustaprotsesse seadme efektiivseks tööks, mis võimaldasid põhifunktsiooni vigadeta täitmist. Taustaprotsesside seas kirjeldasime ära ka kasutajaliidese elemendid ning funktsionaalsuse.

Lõpetuseks kirjeldasime süsteemi paigaldamise eripärad ning analüüsisime koodi leidmise algoritmi reaalsete näitude põhjal. Sellest saime teha järelduse, et süsteem vastab nõuetele, ning leiab alati koodi üheselt valesignaale andmata.

Süsteemi on võimalik edasi arendada lisades teistsuguste tunnustega purkide tuvastamiseks uusi algoritme ning võimaldada nende valimist menüüst. See võimaldaks sama süsteemi kasutada palju rohkemate erinevate toodete asukoha tuvastamiseks.

# OPTICAL POSITIONER FOR THE PRODUCTION OF PAINT CANS

**Hanno Soo**

## Summary

The aim of the project at hand was to design the hardware and software for an optical positioner for the production of paint cans. This system was meant to substitute the currently installed system based on electrical contact.

The project started with a general overview of the design of embedded systems and industrial use of logic controllers. Based on the acquired knowledge system requirements were formed. From there on, the necessary hardware components were analysed and described in depth. In addition, the method to generate a pulse as an output and ways to power various parts of the system were discussed.

A very important part of the design process was to finalize the algorithm to find the specific code based on the input from the sensor. This was thoroughly studied. Next, the timing for output was explained. In addition to the high priority elements of the software it was necessary to run several lesser priority processes. Those include timing, user interface and memory management. Then the proper installation of the system was described.

Finally a further analysis of the code searching algorithm was played through using real data as an example.

# KASUTATUD KIRJANDUS


1. Andres Arrak, "Kas turundus on suur vale?", http://andresarrak.ee/portfolio/kas-turundus-on-suur-vale-delfi-30-05-2013/, (2013)

2. Richard G. Lyons, *Understanding Digital Signal Processing* (Pearson Education, 2010).

3. E. A. Lee and S. A. Seshia, *Introduction to Embedded Systems - A Cyber-Physical Systems Approach* (LeeSeshia.org, 2011).

4. Steve Heath, *Embedded Systems Design* (Newnes, 2002).

5. E. A. Parr, *Industrial Control Handbook* (Industrial Press Inc., 1999).

6. W. Bolton, *Programmable Logic Controllers, Fifth Edition* (Newnes, 2009)

7. Li Tan, *Digital signal processing: fundamentals and applications* (Elsevier/Academic Press, Amsterdam [etc.], 2008).

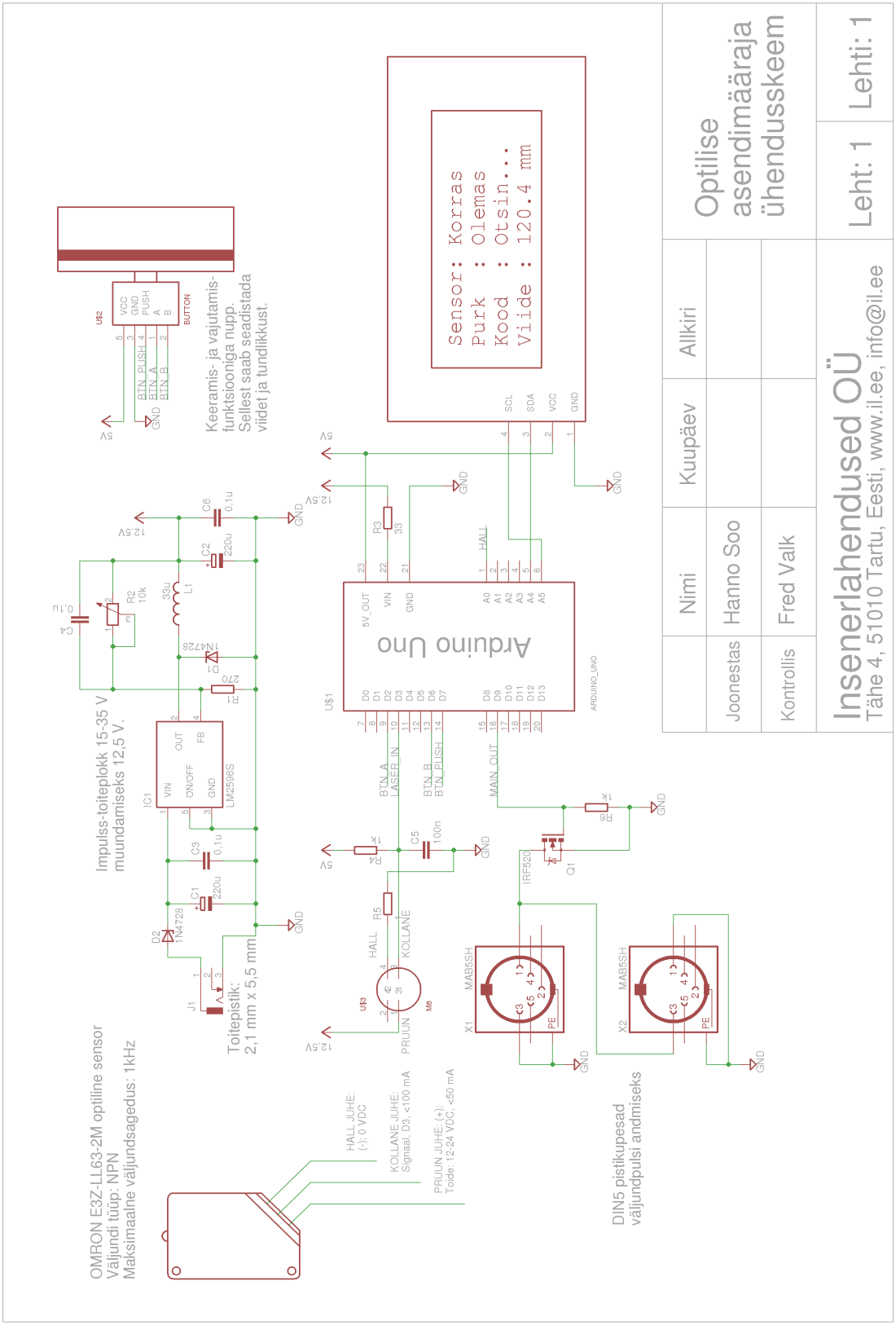
8. E3Z-LT/LR/LL datasheet, http://www.ia.omron.com/data_pdf/data_sheet/e3z-lt_lr_ll_ds_csm2158.pdf, (2014)

9. Thomas Ouellet Fredericks, " Bounce-Arduino-Wiring", https://github.com/thomasfredericks/Bounce-Arduino-Wiring, (2014)

10. Arduino, "LiquidCrystal Library", http://arduino.cc/en/Reference/LiquidCrystal, (2014)

# LISAD

Lisa 1. Mõõtesüsteemi elektriline skeem



Lisa 2. Pilte loodud süsteemist


Pilt 2. Eestvaade. Ekraanil tavaolek töö käigus.

Pilt 3. Vaade küljelt, kus asuvad liidesed.

**Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks**

Mina,_____Hanno Soo_____,
(*autori nimi*)

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) enda loodud teose
_____
_____Optiline värvipurkide asendimääraja_____
_____,
(*lõputöö pealkiri*)

mille juhendaja on _____Fred Valk_____,
(*juhendaja nimi*)

1.1. reprodutseerimiseks säilitamise ja üldsusele kättesaadavaks tegemise eesmärgil, sealhulgas digitaalarhiivi DSpace-is lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;
1.2. üldsusele kättesaadavaks tegemiseks Tartu Ülikooli veebikeskkonna kaudu, sealhulgas digitaalarhiivi DSpace´i kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.

2. olen teadlik, et punktis 1 nimetatud õigused jäävad alles ka autorile.

3. kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest tulenevaid õigusi.

Tartus, **29.05.2014**