TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond
Arvutitehnika instituut

IAF40LT

Gunnar Kotkasets

ATMEGA328 MIKROKONTROLLERIL PÕHINEV UV-A LED LAMBI PROTOTÜÜP GEELLAKI POLÜMERISEERIMISEKS

Bakalaureusetöö

Juhendaja: Margit Aarna
magistrikraad
assistent

Tallinn 2015

Autorideklaratsioon

Olen koostanud antud töö iseseisvalt. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on viidatud. Käesolevat tööd ei ole varem esitatud kaitsmisele kusagil mujal.

Kuupäev: 18.05.2015

Autor: Gunnar Kotkasets

Allkiri:

Annotatsioon

Käesolev bakalaureusetöö keskendub geellaki polümeriseerimiseks vajaliku UV-A LED lambi prototüübi valmistamisele ja tutvustab töö protsessi ideest-valmimiseni. Esimeses etapis kirjeldatakse plaanitava seadme tööprotsess ja tuuakse välja tutvustav tööprotsessi algoritm. Seejärel põhjendatakse mikrokontrolleri valik. Antakse ülevaade kasutajaliidesena kastutatavast TFT LCD ekraanist, funktsioonnuppudest ning automaatselt kontrollitavatest protsessidest. Prototüüpmudel on loodud riistvaralisel tasemel. Viimases etapis esitletakse tarkvara, mis korraldab mikrokontrolleri tööprotsessi ja suudab suhelda andurite, kasutajaliidese: ekraani ning juhtnuppudega.

Antud bakalaureusetöö tutvustab lühidalt Atmega328 mikrokontrollerit, Arduino UNO kloon Iteaduino UNO arendusplaati, HTF0177SN-01 LCD-moodulit. Tarkvara arenduskeskkonnana on kasutatud avatud lähtekoodiga Arduino IDE versioon 1.5.6.2 platvormi. Tarkvara on programmeeritud c++ keeles ning on kasutatud erinevate moodulite avatud lähtekoodiga teeke.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 5 leheküljel, 4 peatükki, 15 joonist, 5 tabelit.

Bakalaureusetöö ülesanne

Üliõpilane: Gunnar Kotkasets, 061654

Lõputöö teema eesti keeles: Atmega328 mikrokontrolleril põhinev UV-A LED lambi prototüüp geellaki polümeriseerimiseks

Lõputöö teema inglise keeles: UV-A LED Lamp On Atmega328 Microcontroller For Gel Lack Polymerisation

Juhendaja: Margit Aarna, Arvutitehnika instituut, tehnikateaduste magister

Lahendatavad küsimused ning lähtetingimused:

UV-A lambi prototüübi valmistamine geel laki polümeriseerimiseks; Atmega328 mikrokontrolleri programmeerimine vajaliku funktsionaalsuse juhtimiseks ning kasutajale informatsiooni edastamiseks TFT LCD-mooduli abil; Prototüübi valmistamiseks Arduino klooni Iteaduino UNO arendusplaadi kasutamine; mikrokontrolleri programmeerimise jaoks keskkonna Arduino IDE versioon 1.5.6.2 kasutamine; toodetud tarkvara mikrokontrollerisse laadimine Arduino IDE abil; TFT LCD-mooduli HTF0177SN-01 kasutamine

Nõuded vormistamisele: Vastavalt Arvutitehnika instituudis kehtivatele nõuetele

Lõputöö esitamise tähtaeg: 08.02.2016

Abstract

UV-A LED Lamp On Atmega328 Microcontroller For Gel Lack Polymerisation

The aim of this paper is to provide workflow for readers of UV-A gel LED lamp prototype on Atmel Atmega328 microcontroller - Introducing all neccecary hardware parts the model is made of. Also source code is described which is particulary written for this microcontroller. This paper is divided into four separate chapters.

In the first chapter the main idea of the UV-A gel LED lamp nature and the actual necessity of this type device is briefly described. Second chapter gives an overview of the particular device construction: author presents the device process worfkflow which is followed by the pre described algorithmic path. Usage and necessary parts of the prototype model is introduced to the reader like Atmel Atmega328 microcontroller and all containing peripheral modules: Iteaduino UNO Development Board, Arduino LCD-module, UV-A LED driver, fan driver, IR-proximity sensor and NTC thermistor sensor. Also Integrated development environment Arduino IDE is presented.

Third chapter is mainly about software development. Arduino Integrated Development Environment and the workflow for developing software MCU is briefly described. The communication between microcontroller's: SPI, Digital I/O, Analouge Input and peripheral's controller ports is described. The main idea of the UV-A LED lamp driver, PWM and on/off signal processing is described through different module elements of the prototype. Finally the developed MCU software is presented.

Fourth chapter sums up and gives reader final overview of the main idea and specific problems that occurred. Author details the main unpredictabilities while writing this paper. Checkpoints as hardware building, algorithm debugging, code debugging were brought out. Author admitted that building reliable hardware and developing bug free code is requiring good understanding manufacturers hardware manuals, needs some know-how and experience playing with electronics.

The thesis is in Estonian and contains 45 pages of text, 4 chapters, 15 figures, 5 tables.

Lühendite ja mõistete sõnastik

Mikrokontroller	*Microcontroller, MCU;* väike integreeritud arvuti ühel kiibil, mis sisaldab protsessori tuuma, mälu elementi, programmeeritavat sisend/väljund ühendusi perifeersete seadmete jaoks.
Valgusdiood	*Light Emitting Diode, LED*; on diood pn-siirdega, mis muudab elektrienergia nähtavaks valguseks või optiliseks kiirguseks nagu infrapuna või ultraviolet spekter.
Pn-siire	*P-N Junction;* Elektrilaengu ülekanne positiivselt negatiivsele monokristall pooljuht alale.
Diood	*Diode*; Elektroonika element, millel on ühesuunaline elektrijuhtivus.
Kompaktluminofoorlamp	*Compact Fluorescent Lamp, CFL*; Väikese mõõtmeline koduseks kasutamiseks mõeldud luminofoor lamp, mis on konstruktsioonilt mõeldud püsima vigastuste eest terved.. Sisaldab enamasti individuaalset voolutrafot toite jaoks.
Kärbitud käsustikuga arvuti	*Reduced Instruction Set Computing, RISC;* mikroprotsessori arhitektuur, mille töökäsustik on optimiseeritud väiksemale arvule suurema töökiiruse saavutamiseks. Kiibist vabaks jäänud füüsilist ala kasutatakse vajalike loogika elementide käskude kiirendamiseks. Programmid on suuremad ja võtab rohkem mälu, sest enamus tööd emuleeritakse tarkvaraliselt .
Komplekse käsustikuga arvuti	*Complex instruction Set Computer, CISC;* Arhitektuurilt keerulisem mikroprotsessor, mis võib ühe käsu raames teha mitmeid erinevaid mikrooperatsioone. Masinkoodis lihtne kirjutada ja tänu keerukale loogikale võtab kood nii vähem ruumi kui mälu.

Analoog-digitaal konverter	*Analog To Digital Converter, ADC, A/D;* Muundur mille abil muudetakse analoog signaal digitaalseks ja salvestatakse või opereeritakse binaarse infona.
Välkmälu	*Flash;* Mälu andmete või programmide salvestamiseks, mida saab elektriliselt ümber programmeerida ja kustutada väikeste plokkide kaupa.
Operatiivmälu	*Random Access Memory, RAM;* Muutmälu, mis on välkmälust tunduvalt kiirem. Kasutatakse puhver mäluna oma kiiruse pärast protsessori operatsioonide ning püsimälu vaheliseks suhtluseks.
Taktsagedus	*Clock Rate;* Taktgeneraatori poolt genereeritav impulss, mida kasutatakse arvuti protsessori ja teiste perifeersete seadmete sisese ja omavahelise suhtluse sünkroonis hoidmiseks.
Ostsillaator	*Oscillator; G*eneraator, mis tekitab sumbumatuid elektrilisi harmoonilisi võnkumisi taktsagedus ignaali genereerimiseks.
Watchdog taimer	*Watchdog timer, WDT;* Taimer mille eesmärk on tuvastada arvuti töös hälbeid. Enamasti kasutatakse sardsüsteemides. Seadme hangumise või mõne muu tõrke puhul taaskäivitab süsteemi või selle osa.
Käsku sekundis	*Instructions per second, IPS;* Arvuti protsessori kiiruse mõõtühik. Näiteks MIPS on (*million instructions per second*) ehk million käsku sekundis.
TFT Vedelkristallkuvar	*Thin Film Transistor Liquid Crystal Display, TFT LCD;* Vedelkristall kuvari tüüp, mis kasutab õhukeste kilede transistorite(TFT) tehnoloogiat.
Integreeritud arenduskeskkond	*Integrated Development Environment, IDE;* Tarkvara rakendus, mis integreerib endas kõiki vajalike programmide arendamiseks vajalike funktsionaalsusi spetsiifilise või mitmete programmeerimiskeelte tarbeks.
Polümerisatsioon	*Polymerization;* Ehk polümeriseerumine on keemiline protsess, milles madalmolekulaarse ehk monomeeri

	ühendid ühinevad üksteisega moodustades makromolekulaarse ühendi.
Ultraviolet kiirgus	*Ultraviolet radiation, UV;* Elektromagnetkiirgus, mille lainepikkus on väiksem kui nähtav lainepikkus. UV-A lainepikkus on 315-380 nanomeetrit.
Pulsslaiusmodulatsioon	*Pulse With Modulation, PWM;* Elektrilise signaali pulseerimine pinge sisse ja välja lülitamise teel. Analoogsignaali matkimine digitaalväljundites signaali juhtimiseks.
SD mälukaart	*Secure Digital, SD;* Püsimälu, mida kasutatakse põhiliselt kaasaskantavate või väike seadmete informatsiooni talletamiseks.
Nanomeeter	*Nanometre, nm;* SI ühik, mis on $1{,}0 * 10^{-9}$ meetrit
Infrapuna kiirgus	*Infrared, IR;* Elektromagnetkiirgus, mille lainepikkus on nähtava lainepikkuse ja nn mikrolainepikkuse piiripeale jääv kiirgus. 700nm – 1mm.
Termistor	*Thermistor, NTC;* Termotakisti on termoelektriline seade mille takistus muutub mittelineaarselt vastavalt temperatuurikõikumisele.
Järjestik kommunikatsioon liides	*Serial Peripheral Interface, SPI;* Sünkroonne lühi maa suhtlemis liides sardsüsteemides.
MISO	*Master Input, Slave Output;* SPI liidese signaali andmeviik juhitavalt seadmelt kontrollerisse.
MOSI	*Master Output, Slave Input;* SPI liidese signaali andmeviik juhtivalt seadmelt juhitavasse seadmesse.
SCK/SCLK	*Serial Clock;* Juhtiva ja juhitava seadme sünkroonimissignaal.
CS/SS	*Slave Select;* Juhitavate seadme valik signaal järjestik ühenduses olevate valimiseks juhtiva seadme poolt.
Täis dupleks	*Full-duplex*; Suhtlemis süsteem, kus seadmed saavad teineteisega kommunikeeruda samaaegselt, nagu näiteks kaks telefoni, mis saadavad mõlema osapoole häälkõne samaaegselt teine teises suunas.

Akrüül	*Acrylic;* Keemiline ühend, mis sisaldavad akrüülrühmadest pärit akrüülhappeid.
Monomeer	*Monomer; M*olekul mis on võimeline siduma keemiliselt teisi molekule polümeerideks.
Oligomeer	*Oligomer;* Molekulaarne kompleks, mis sisaldab erinevate monomeeride allrühmi.
Isoleeritud paisuga välja transistor	*Metal Oxide Semiconductor Field Efect Transistor, MOSFET;* Väljatransistor, milles elektriväljaga muudetakse laengukandjate konsentratsiooni kanalis.
Kvant paisuga välja transistor	*Quantum fielt effect transistor, QFET;* Väljatransistor, mille eeliseks on kvant tunneldamine, et suurendada kiirust traditsiooniliste elektron juhtivusega transistorite ees.
Transistor	*Transistor;* Kolme väljaviiguga pooljuht elektriahelate lülitamiseks ja elektrisignaalide võimendamiseks.
Emitter	*Emitter;* Transistori välja viik võimendatud signaali sisenemiseks
Kollektor	*Collector;* Transistori väljaviik võimendatud signaali väljumiseks
Baas	*Base; Transistori* sisend viik võimendatud signaali lülitamiseks lülitamiseks sellele pinge peale andmisel.
Tahhomeeter	*Tachometer timer, Tach;* Mõõteriist pöörlevate masinaosade nurkkiiruse mõõtmiseks.
Darlingtoni transistor	*Darlington transistor;* Kahe bipolaarse transistori paar integreeritud skeemis suurema signaali võimenduse saamiseks ruumalakokkuhoiuga trükplaadil.
Lüliti	*Switch, ON-OFF;* elektroonika ahelas kasutatav elektriahela ühendaja või katkestaja.
Anood	*Anode;* Elektriseadme elektrood, millele liiguvad katoodilt väljuvad negatiivse elektrilaengu kandjad.
Katood	Cathode; Elektriseadme elektrood, millelt väljuvad negatiivse elektrilaengu kandjad.
Elektrood	*Electrode;* Elektrijuht, mida kasutatakse mittemetallilise kesskonna või kehaga kokkupuuteks nii elektrilise

	ühenduse loomiseks kui ka keemilistes protsessides osalemiseks voolujuhtidena.
Universaalne järjestisiin	*Universal Serial Bus, USB;* Arvuti järjestik siini ühendusstandart välisseadmetega suhtlemiseks.
Sardsüsteem	*Embedded system;* Arvuti süsteem kitsafunktsionaalsusega töö tegemiseks, mille kõike elektrilised ahelad on integreeritud ühtse trükkplaadi peale.
Alglaadur	*Bootloader;* Programm mikrokontrolleril, mis lubab uue püsivara laadimist mikrokontrollerisse läbi välise paigaldus keskkonna nagu Arduino IDE USB ühendus. Ilma selleta saab ainult läbi TX/RX portide püsivara laadida otse mikrokontrolleri vastavatele jalgadele füüsilised ühendused teha püsivara laadimis riistvaraga.
Püsivara	*Firmware;*On seadme riistvaraline programm, kus on tarkvaraliselt määratud riistvara poolt tehtavad funktsioonid ja operatsioonid. Püsivara ei saa opereerimise ajal muuta, see on nagu riistvara funktsioonide instruktsioonide andmed.

Sisukord

Jooniste nimekiri

Tabelite nimekiri

1. Sissejuhatus

Eksimine on inimlik; samuti on inimesele keeruline meelte abil ajavahemike hindamine rääkimata täppis mõõtmisest. Kõigi inimlike puuduste korvamiseks oleme loonud juba tsivilisatsiooni algusest peale endale abivahendeid. Esimene kivist tööriist on teadaolevalt pärit 2.5[19] miljoni aasta tagusest ajast. Esimene mehaaniline kell aja täpsemaks hindamiseks asendas õige pea veekella[20], mis omakorda on päikesekella järeltulija. Kuigi tehnoloogia areng pole ka veel tänaseks peatunud on nüüdisajal meie elu keskseks osaks mikrokontrollerid, mille võidukäik sai alguse peale elektroonika tehnoloogia kolimist pooljuhtidele.

Mikrokontroller on isemõtlev seade oma ette programmeeritud töökorralduste automaatseks täitmiseks. Võib nimetada ka mikroarvutiks. Tema ülesanne on võtta vastu infot kõikvõimalikelt sisenditelt nagu sensorid või kasutajaliides ning töödelda seda. Mikrokontrollerist on saanud paratamatu mikroelektroonikasüsteemi osa, mis peab registrit iga tema kontrolli all oleva seadme üle. Väike kuid võimas mikroarvuti kontrollib aega, mõõdab sisendite signaali ja annab väljund seadmetele vastavaid korraldusi, suhtleb kasutaja liidesega ning vajadusel ka teiste mikrokontrolleritega vastavalt oma paiknemisele hierarhilises süsteemis.

Kontrollerite kasutamiseks on mitmeid eeliseid: tsentraalne süsteemi kontrollitus, täpsus, inimesele raskete või võimatute ülesannete ülevõtmine ning antud inimressursi üle üldine vähene sekkumise vajadus.

Esimene mikroprotsessor on pärit 1971. aastast. Intel sai valmis mudeliga 4004, mis omas 4-bitist andmetöötlus mahtu, vajades funktsioneerimiseks küll mõningate teiste veel integreerimata loogika elementide abi teiste kiipide näol. Lühidalt peale seda 1974 tuli ka kommerts kasutusse esimene integreeritud kiip: protsessori, taktgeneraatori[17], püsi-ja muutmäluga – koos kõigi sisend-väljund viikudega[1].

Antud bakalaureuse töö ülesandeks on luua UV-A[21] LED lambi prototüüp. Seadme südameks on Atmel Atmega328 mikrokontroller[22], millega tutvume järgnevas töös lühidalt ka järgmistes alapunktidest – nii riistvaralises, kui ka programmeeritava tarkvara funktsionaalsuse osas.

Seadme eesmärk on polümeriseerida akrüüli koostisosadel põhinevat ainet lambi poolt kiiratava UV-A kiirgusspektri ehk nn musta valguse abil[4], [44]. Antud juhul on tegemist geellakiga, mis kaunistab ja eelkõige kaitseb naise pikki küüsi igapäeva elus, mille sisse on lisatud pigmenti andes lakile soovitava värvi ning läike. Vajadus uue põlvkonna UV-A lambi järele on suur, kuna järjest populaarsemaks muutuvad LED[2]-[3],[45] valgus allikad vahetavad välja CFL[5] valgus allikad. Geellaki tehnoloogia arenguga on saanud võimalikuks laki kiirem kivistumine ning LEDid kiirgavad omaduste poolest kitsamat lainepikkust, mida saab kohaldada konkreetsete turul toodetavate lakkidega. Tulemusena on kivistumine kiirem, lakk valgub peale kandmisel vähem laiali ja enne kivistumist saab protsessi paremini kontrollida korrektsema tulemuse saavutamiseks.

Prototüübina valmiv UV-A lamp kontrollib taimeriga kasutaja poolt valitud aja jooksul UV-A LEDide sisse ja välja lülitamist. Jahutab LEDide poolt toodetavat kuumust ventilaatoriga, kui termoandur tuvastab etteantud temperatuuri ületamise piiri. Ventilaatori pöörlemise sagedus sõltub temperatuuri kõrgusest. Prototüübi versiooni edasi arendades on LEDide tugevust võimalik vajadusel PWM[6]-[7] meetodiga määrata. Samuti saab täiustada menüüd, muutes selle atraktiivsemaks ja kasutajasõbralikumaks. Võimalik, et menüü värve ja teksti suurust saab muuta. Ja miks ka mitte puutetundliku ekraani lisamisel funktsioonnupud ekraanile viia.

Prototüübi edasiarendamise suunad sõltuvad otseselt testide ja katsete tulemusest. Peale katseandmete kogumist saab geellaki vajalikuks polümeriseerimiseks kindlaks teha LEDide vajaliku intensiivsuse astme. Temperatuuri optimaalsena hoidmiseks saab samuti peale andmete kogumist korrigeerida ventilaatori ja aktiivjahutuse suhestumist jahutatavate moodulitega. Kõige esimeseks edasiarendus plaaniks näen logi faili kirjutamise võimalust micro-SD kaardile, mis kogub andmeid anduritelt ja jälgib mõningat lõppkasutaja tegevuse mustrit.

Üheks tähtsaimaks protüübi edasi arendamise eeliseks näen universaalse UV-A LED lambi tootmist. Spetsiifiliste LEDide kiirgusvahemik on väga kitsas – kuskil 10nm. Geelid mida täna toodetakse sisaldavad fotoinitsiaatoreid[25] enamasti 340-380nm lainepikkusele[38]. Seega lampi saab tulevikus koostada erinevate kitsaste kiirgusvahemikega LEDidega ja luua profiilid vastavate geelide jaoks, milles tuleb mängu mikrokontrolleri tõeline kasumlikus.

2. Ülevaade prototüüpmudelist

Käsitletav peatükk peatub UV-A lambi prototüüpmudeli algoritmilisel töövoo tasemel; annab ülevaate erinevatest kasutatavatest elementidest ja põhjendab nende valikuid; peatub lühidalt mikrokontrolleritel ning tutvustab isendit Atmel Atmega328[22]; tutvustab Arduino UNO kloon Iteaduino UNO arendusplaati[26]-[27]; annab ülevaate perifeersetest seadmetest: UV-A 370-380nm 3W High Power LED[29], TFT LCD moodul HTF0177SN-01[31], Sunon MagLev ME40101V1-000U-G99[32] ventilaator; selgitab elementide omavahelist ühendamise loogikat ja suhtlemist; esitleb reaalselt valmis ehitatud prototüüpmudelit koos tarkvara ja selle programmeerimiskeskkonnaga.

2.1. Mudeli töövoo algoritm

Seade on mõeldud peale prototüübi välja arendamist kolmandatele osapooltele tarbimiseks. Kuna seadme kasutaja ei pea olema tehniliste teadmistega, siis lõpptulemus peab olema intuitiivselt kasutatav. Selleks, et seadme töös ei tekiks tõrkeid ja ettearvamatuid olukordi tuleb kirjeldada seadme töövoo algoritm. Juhtelementide töövoog peab olema selgelt kirjeldatud ja arusaadav seadme kokku monteerimiseks ja tarkvara arendamiseks. Lõpp produkti tuumikuks on mikrokontroller, mille töö algoritm kirjeldatakse antud peatükis ka kõigepealt ära. Algoritmi koostamiseks on vaja ka tähtsamate perifeersete seadmete mainimist lisaks mikrokontrollerile – LCD-moodul, juhtnupud, LED kiirgusallikad, andurid nagu seadme sisselülitamise andur käe sisestamisel masinasse ja termoandur LEDide aktiivjahutuse sisse lülitamiseks. Kõikide komponentide ülesanded kirjeldatakse lisaks eelpool tähtsamatele mainitud elementidele peatüki arenedes. Seda kõike kontrollib ühtne mikrokontroller vastavalt järgnevalt algoritmis kirjeldatud tingimustele. Tööülesandeks on maniküürsüsteemides kasutatava geellaki mono-ja oligomeeride polümeriseerimine[36] ehk tinglikult kivistamine – kui nii võib öelda.

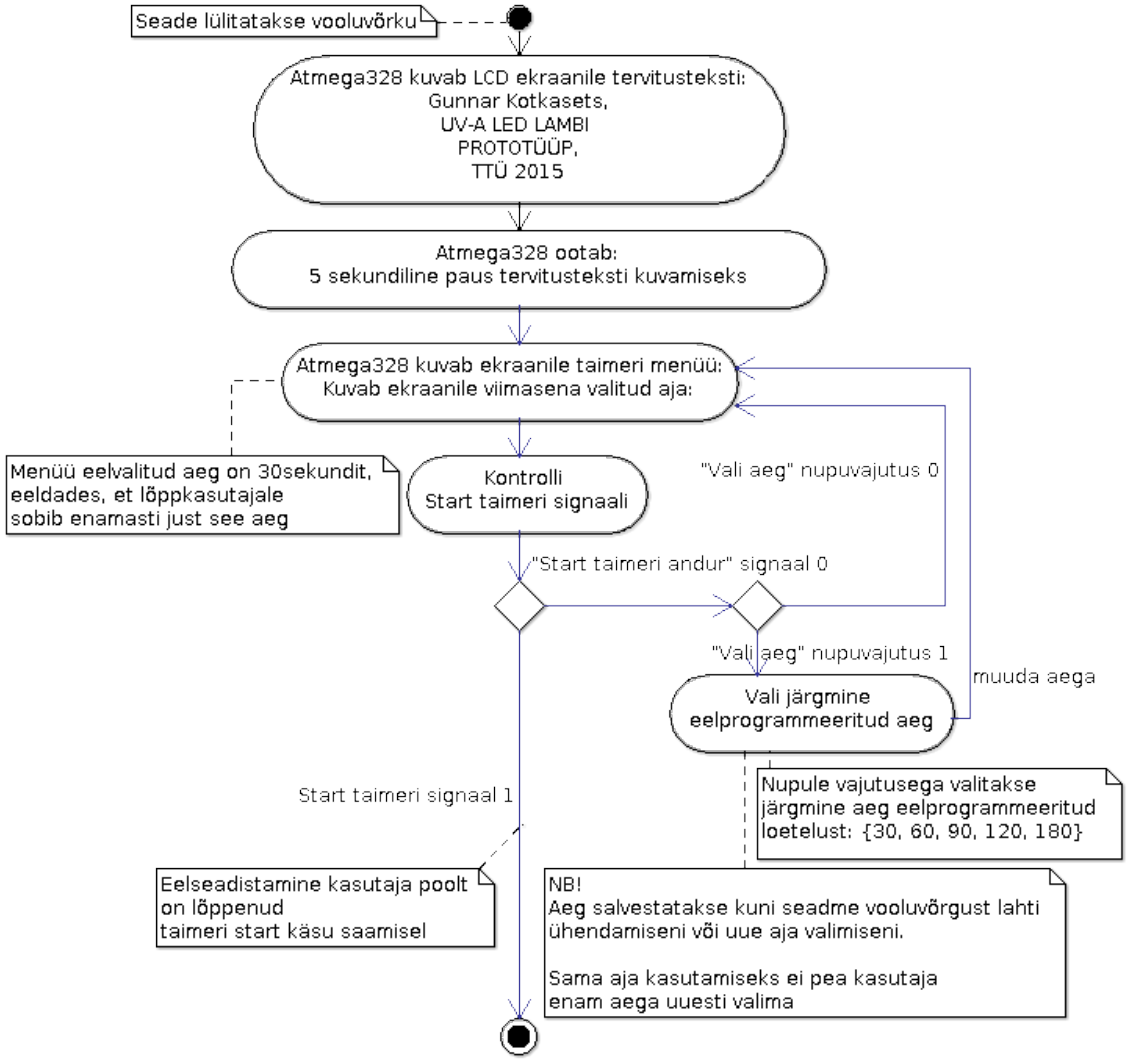
Doug Schoon (Scoon Scientific LLC president) video sarjade („Face-To-Face with *Doug Schoon*”) eesti keelsele kokkuvõtvale artiklile[36]-[37] põhinedes on olemas kaks põhilist UV geel maniküürsüsteemi, mida kasutatakse küünte ja kunstküünte kattematerjalideks. Esimeseks tüübiks on modifikatsioonid nn „pehmetest” UV

geelidest, mis on nn „kõvade” UV geelide järeltulijad. Teine tüüp UV geellakkidest on nn „hübriidgeelid”. Kahe tüübi erinevuseks on lahustitega eemaldamise aeg, kuid ühisosaks on polümeriseerimine UV-A lambiga. Prototüüp on suunatud geellakkidele, mille fotoinitsiaatorid aktiveeruvad 370-380nm UV-A lainepikkusega kokku puutudes[38]. Ideaalis 375nm lainepikkusel olev fotoinitsiaator kivistab geeli 30sekundiga. Siiski vahemärkusena lisades võib erinevatest toodetest leida polümeriseerimist aktiveerivaid fotoinitsiaatoreid vahemikus 340-380nm[39].

Mudeli juhtelementideks on mikrokontroller, LCD-moodul, menüü nupp, taimeri aktiveerimise andur. Mudeli eesmärgiks on lülitada sisse LED allikad taimeriga etteantud ajaks, et geellaki sees olevad fotoinitsiaatorid, saaks kiirguse abil läbi viia aine polümeriseerumus protsessi. Taimer aktiveeritakse IR-lähedusanduriga[40], mille käivitus signaal saadakse IR LEDilt käepeegeldumisest tekkinud signaali muutusega IR-fototransistoris. Taimeri loendaja nulli jõudes lülitatakse LEDid välja.

Enne taimeri aktiveerimist on kasutajal „Vali aeg” nupuga võimalik muuta lambi töörežiimi pikkust. Erinevad ajalised kestvused on eelprogrammeeritud ning antud valik on tehtud geellakkide pakendite peal oleva enim mainitud variantidega.

Kui lamp on juba töörežiimis ehk peale LEDide sisselülitumist hakkab mikrokontroller kohe kontrollima NTC termoanduri[41]-[42] temperatuuri, mis asub LEDide passiiv jahutuse külge kinnitatuna. Ventilaatori sisse-väljalülitumist ja pöörete kiirust reguleeritakse ja kontrollitakse peale igat tsükli algust.

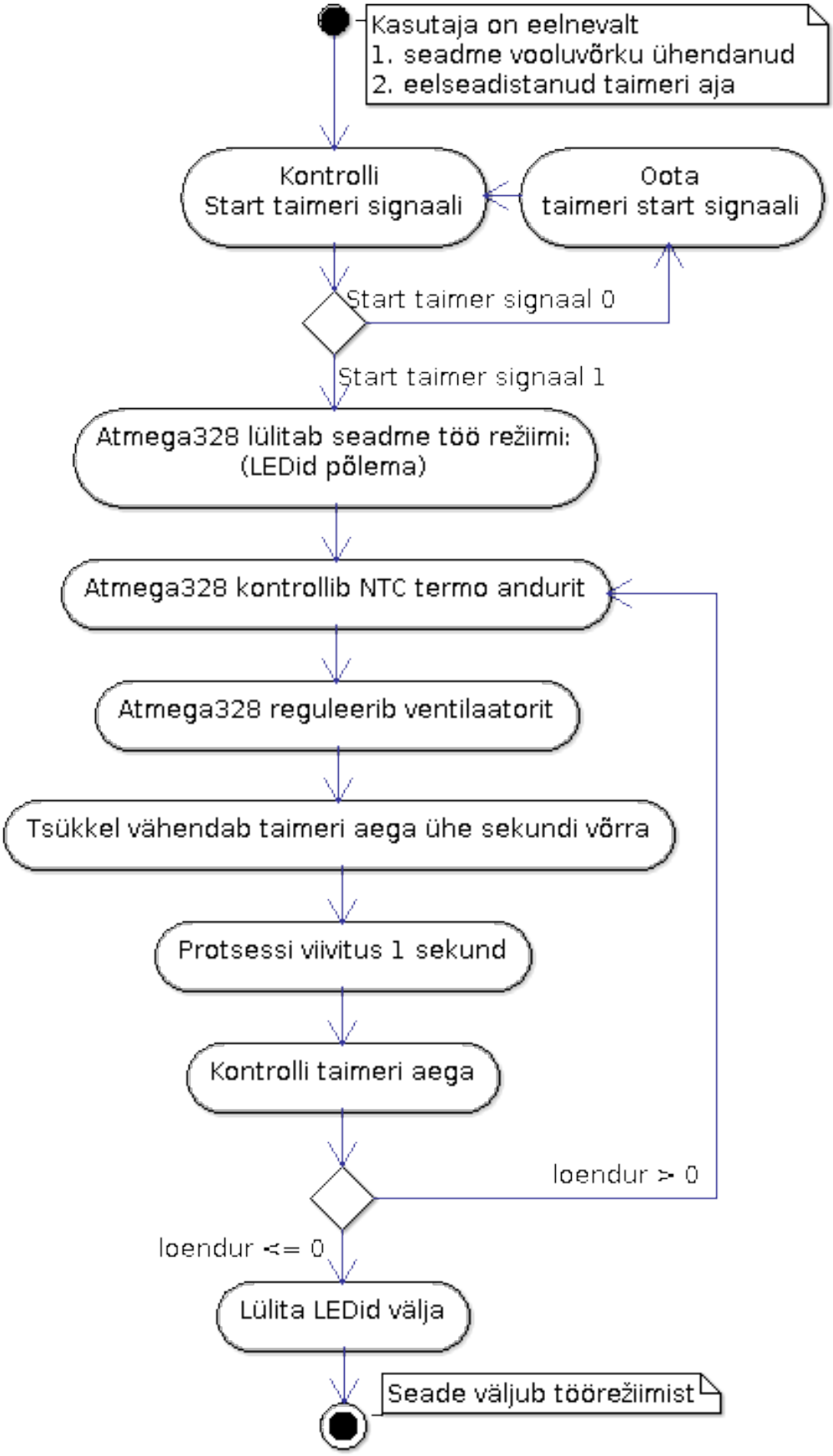


Joonis 2.1: Mudeli vooluvõrku lülitamise ja eelseadistamise tegevusalgoritm

Järgnevalt on välja trükitud neli prototüüpmudeli tegevusalgoritmi joonist. Esimene (Joonis 2.1) keskendub seadme sisse lülitamisele ja kasutaja menüü kuvamisele, kuni eelseadistamisest väljumiseni. Teises tegevusalgoritmis (Joonis 2.2) on loogika peale menüü eelseadistamist, kus sisenetakse lambi mudeli töörežiimi. Töörežiimi alguses lülitatakse LEDid sisse; pärast seda kontrollitakse NTC termoanduri abil temperatuuri passiivjahutuse radiaatorilt; reguleeritakse vajadusel väljatõmbe ventilaatori töö kiirust; vähendatakse loenduri aega ühe sekundi võrra; kontrollitakse, kas loendur on jõudnud nulli ja kui pole korratakse tsüklit algusest peale. Loenduri null jõudmisel väljutakse töörežiimist.

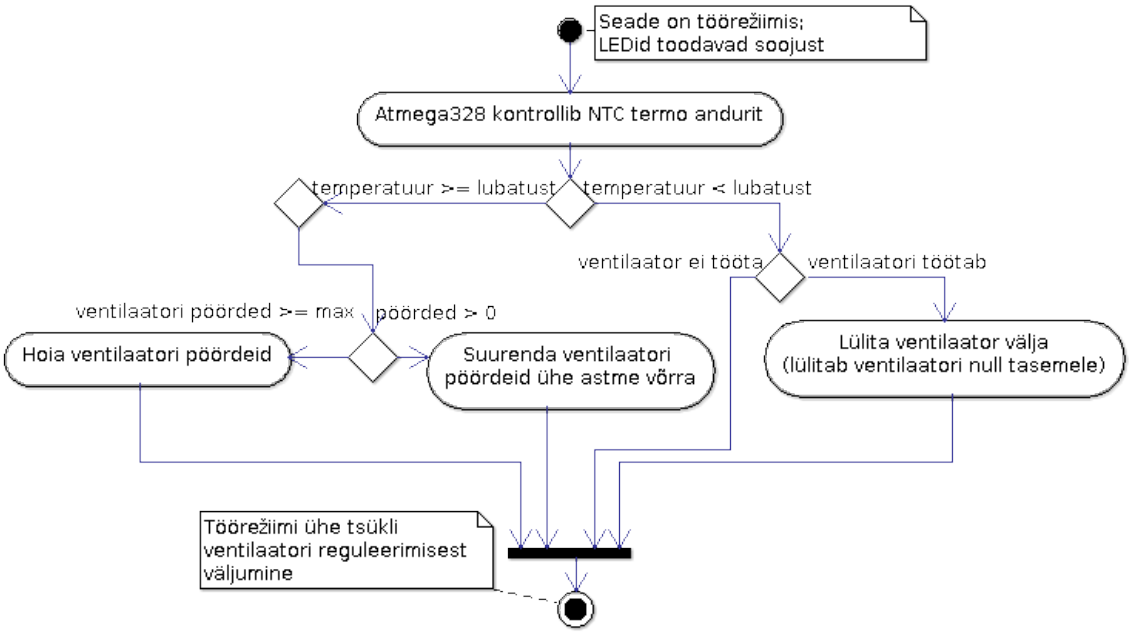
Kolmanda ja neljanda tegevusalgoritmi (Joonis 2.3 ja 2.4) puhul on tegemist aktiivjahutuse loogikaga vastavalt töörežiimis ja peale seda. LED kiirgusallikate põlemise ajal kontrollitakse iga loenduri vähendatud sekundi ees pidevalt passiv

jahutuse suutlikust hoida temperatuur optimaalsena, kuna teatavasti valgusdioodid toodavad üsna palju soojust.

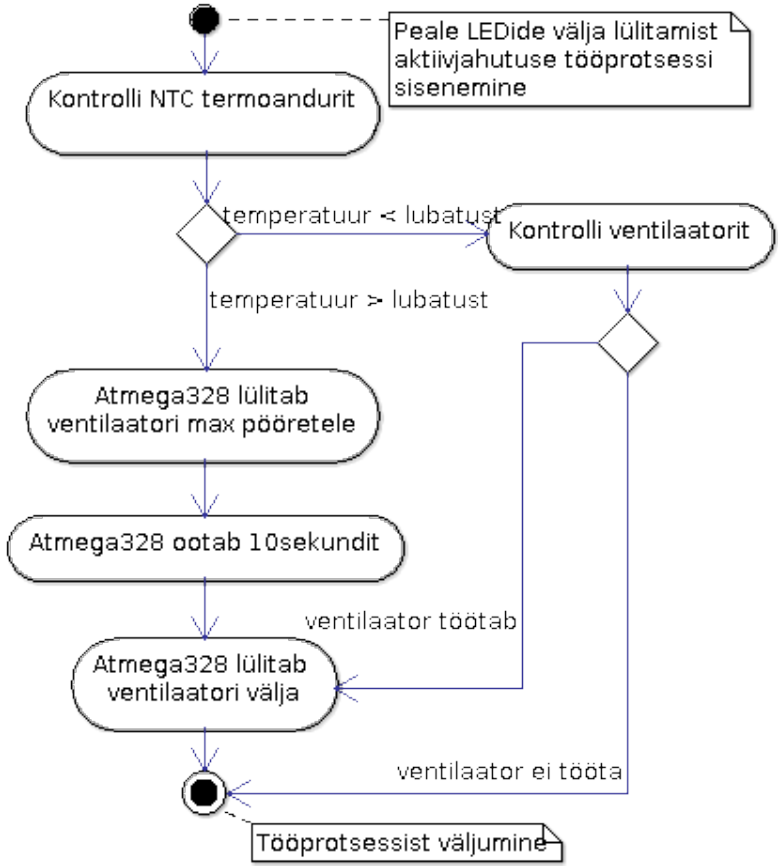


Joonis 2.2: Mudeli töörežiimi tegevusalgoritm

Teatud juhtudel võib olla vajalik sekkuda aktiivjahutusel ka peale valgusdioodide väljalülitumist abistavalt passiiv jahutuse töö tõhustamiseks tööprotsessi. Selleks on lisatud täiendav temperatuuri kontroll töörežiimist väljumisel ning vajadusel lülitatakse ventilaator kümneks sekundiks maksimaalsetel pööretel sisse.



Joonis 2.3: Mudeli aktiivjahutuse tegevusalgoritm seadme töörežiimis



Joonis 2.4: Mudeli aktiivjahutuse tegevusalgoritm peale töörežiimi lõppu

Täielik Prototüüpmudeli tegevusalgoritm on toodud lisas (Lisa 1).

2.2. Kasutatud mudeli detailid

Prototüüpmudeli koostamise jaoks vajalik põhiliste elementide olemasolu:

1. Mikrokontroller
2. Iteaduino UNO arendusplaat
3. LCD-moodul
4. UV-A LEDid
5. Ventilaator
6. Toiteadapter
7. Reaalne prototüüpmudel ja vajalikud aksessuaarid

Elementide valiku põhjendus ja nende iseloomustus on kirjeldatud järgnevates alampunktides. Vahemärkusena on eelistatud autoril juba olemasolevaid elemente, kuna aitab vähendada rahalisi kulutusi. Siiski teatud elemendid nõuavad oma eripära tõttu rahalisi väljaminekuid.

2.2.1. Mikrokontroller Atmel Atmega328

Mikrokontrolleriks on valitud Atmel Atmega328[22]-[24], kuna järgnevalt tutvustatav arendusplatvorm (Iteaduino UNO[26]-[28]) omas seda integreeritult. Samas on see mikroprotsessor üks kaasaegsemaid ja oma omaduste poolest optimaalne käesoleva prototüüpmudeli valmistamiseks nii arvutusvõimsuse kui sisend-väljund viikude poolest. Eeliseks võib välja tuua veel seda, et antud mikroprotsessori tarkvara arendusplatvormil on rikkalik kasutaja tugi ja peale selle saab seda hiljem piltlikult teise platvormi ehk eraldi seisva mikroskeemi peale tõsta masstootmiseks vajaliku disaini ja optimaalsusega.

Mikrokontroller sisaldab juba vajalike eraldiseisvaid detaile: keskprotsessorit, ostsillaatorit, analoog-digitaal konverterit[13], püsimälu, muutmälu ja teisi vajalike loogikaelemente. Samuti on mikrokontroller peale integreeritud seadme eelistele ka piisavalt võimas, et kompaktsus, mõõtmed ja tema odavus kaaluvad üle mikroprotsessori eraldi kasutamise koos eelpool mainitud abi loogikaelementidega eraldiseisval emaplaadil. Sellistele kriteeriumitele lähtudes saame hinnalt odavaima, suuruselt kõige mõistlikuma ja võimsuselt optimaalseima seadme südameks just nimelt Atmega328 oma paljude kaasvõitlejate seast antud seadme valmistamiseks. Tabelis 2.1 on välja toodud antud mikrokontrolleri tähtsamad omadused[22]-[24].

Atmega328 mikrokontroller kuulub Atmeli mega-AVR sarja seeriasse tema südameks 8-bitine AVR RISC arhitektuuril baseeruv mikroprotsessor. RISC[8]-[10] arhitektuur on disainitud olema kiire, lihtsakoelise riistvara ja lühema arendustsükliga. RISCi muudab kiireks lihtsustatud instrueeritud käsustiku süsteem, mis lubab andmete kiiremat töötlemist seeriatesse ühendatud siinide kaudu parallelismi teel ja muudab andmetöötluse CISC arhitektuuri[11] ees 2-4 korda kiiremaks. Lihtsustatud CPU käsustiku teel muutub ka riistvara pindala väiksemaks kiibil, mis lubab lisada ekstra funktsioone. Samale kiibile saab lisada veel mäluhalduse, ujuvkomaarvutuse ja aritmeetika loogikaplokid. Samuti on RISC arhitektuuril baseeruvad protsessorid ka lühema arendustsükli ajaga, kuna lihtsat loogikat on võimalik kiiresti ümber konverterida kiirestiarenevale tehnoloogiale võrreldes CISC riistvara keerukusega. Kõik see muudab ühe süsteemi ka energiasõbralikumaks.

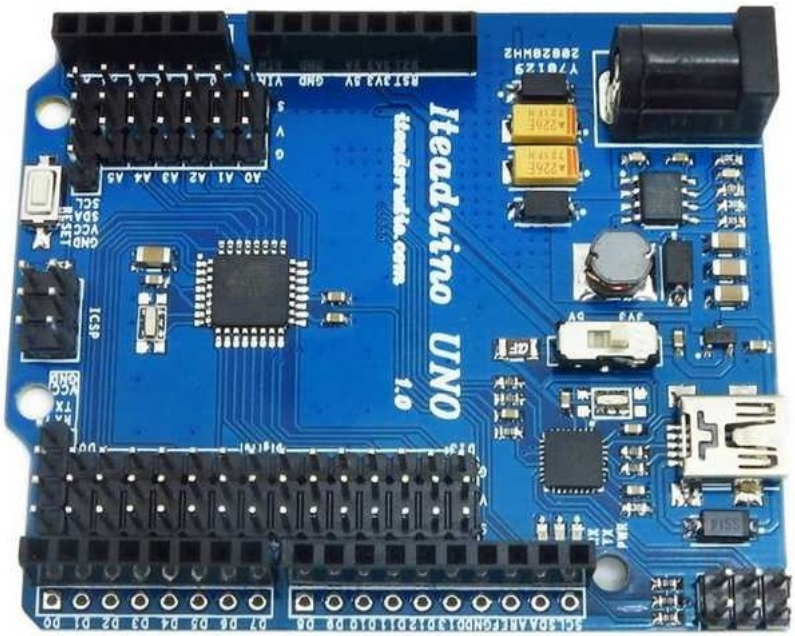
Tabel 2.1: Atmel Atmega328 mikrokiibi karakteristik.

Atmel Atmega328 funktsionaalsus	
Programmi mälu (Flash)[14]	32 kB
Muutmälu (RAM)[15]	2 kB
Sisend-väljund viikude arv	28 tk
Taktsagedus (Maksimum)[16]	20 MHz
Protsessor	8-bit AVR
Registrid	32 tk
Taimerid/Loendajad	3
Sisend-väljund katkestid	Jah
Järjestikkommunikatsioonid	USART, SPI
A/D konverter	6-kanaliga 10-bit
Watchdog taimer[18]	Programmeeritav koos sisemise ostsillaatoriga
Pinge vahemik opereerimiseks	1,8-5,5V
IPS[12]	1 MIPS/MHz

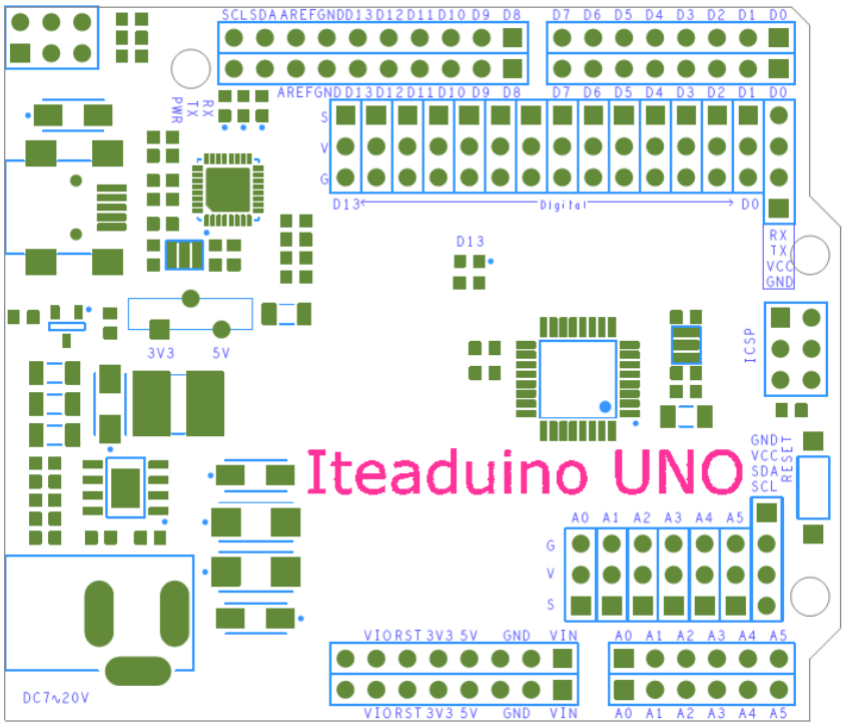
2.2.2. Iteaduino UNO arendusplaat

Arendusplaadina on kasutatud Arduino UNO[46] klooni Iteaduino UNOt[26]. Mikroprotsessori funktsionaalsuse arendamine valmis oleva arendusplatvormi peal on üsna loogiline valik. Kogu arendamiseks vajalik mikroelektroonika on kohe olemas; ei ole vaja lisaaega arenduseks vajaliku skeemi juurutamiseks; lisakulutused vajaliku keskkonna loomiseks kõrgemal tasemel disaini poolelt on juba tehtud. Iteaduino valikul oli Arduino ees eelisteks odavam hind, kohe poes olemas ja 3.3V loogika võimalus. Miinuseks Arduino ees, aga mikrokontrolleri integreeritus plaadil. Siiski pole see määrav, kuna mikrokontrolleri enda hind on nagunii nii madal, et eralditöötava maketi peale tõstes oleks nagunii vaja teist kiipi, sest muidu jääks üks lihtsalt ilma. Arendusplaadi illustratsioonidega on võimalik tutvuda Joonistel 2.5 ja 2.6

Arendusplaadile on mugavalt välja toodud mikroprotsessori sisend-väljund viigud. Antud töös kasutatakse A0-5 porte sensorite andmete vastuvõtuks ja D0-13 perifeersete seadmete juhtimiseks ning funktsioonnuppude sisenditeks.



Joonis 2.5: Iteaduino arendusplaadi illustratsioon



Joonis 2.6: Iteaduino arendusplaadi ühenduste illustratsioon

2.2.3. LCD-moodul

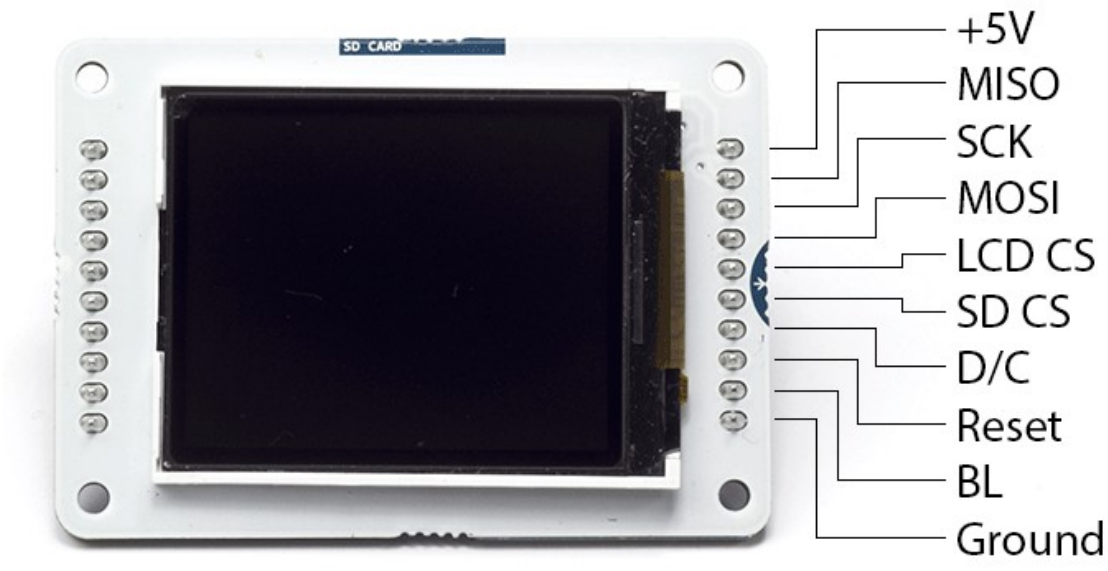
TFT LCD-moodul HTF0177SN-01[30] valisin edevuse ja mugavuse poolest. Edevaks teeb selle värviline ekraan, mis suudab soovi korral ka värvilisi pilte kuvada. Mugavaks teeb antud perifeerse seadme see, et tal on Arduino ühilduvusega makettplaat juba taoliste arenduste jaoks külge disainitud koos micro-SD kaardi lugemise mooduliga. Mugavalt on olemas ka avatudlähte koodi teegid[33], mistõttu võib LCD-moodulile info kuvamiseks kasutada olemasolevaid meetodeid. Tabelis 2.2 tähtsamad TFT LCD-mooduli omadused[31].

Tabel 2.2: TFT LCD-moodul HTF0177SN-01 karakteristik

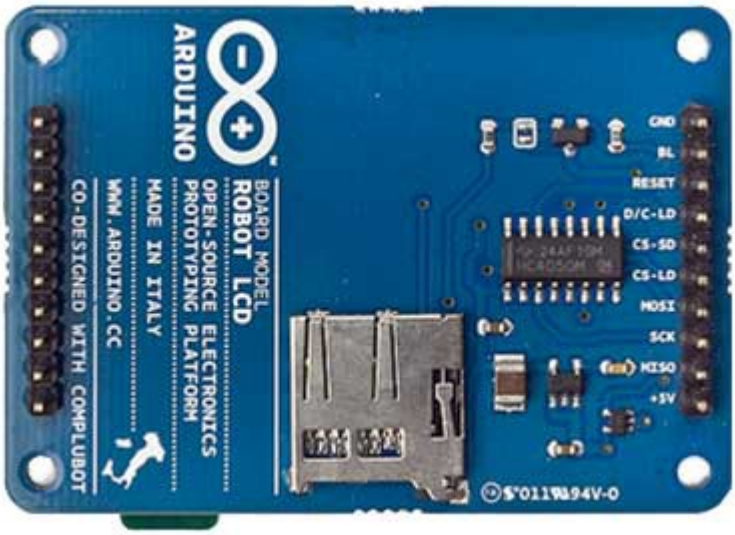
HTF0177SN-01 omadused	
LCD tüüp	1,77 tolli TFT LCD
Värve	262 000 erinevat
Nägemise nurk	12:00
Aktiivne ala (L*K)	28,032*35,04mm

Punktide arv	128(R,G,B)*160
Kontrolleri mudel	ILI9163
Ekraani toitepinge	2,7~3,3V
Paneeli sisend pinge	5V
Tagataustavalgus	Valge LED
Opereerimise temperatuur	-20~+70°C
Andmeedastus	4 realine seeria

Antud LCD ekraani juhitakse SPI[34]-[35] (Serial Peripheral Interface) sünkroonse järjestik kommunikatsioon liidese kaudu. Mõnikord kutsutakse ka 4 juhtme järjestik liides ühenduseks. Tüüpiline andmete edastus kahe osapoole vahel käib läbi nihke registri, kus mikrokontroller saadab väljundi MOSI ja LCD ekraan MISO siini pidi. Sünkroonivat signaali saadab SCLK siin. Antud juhul on LCD ekraaniga ühel makettplaadil ka SD-kaardi lugeja, seega kuna mikrokontrollereid on üks, siis valitakse LCD/SD CS siinide abil kumma seadmega tahetakse infot jagada või vastu võtta. Osapoolte vaheline suhtlus toimub kõik *full-duplex* tasandil, ehk nii andmeid saab saata samaaegselt mõlemas suunas. Joonistel 2.7-2.8 ja on toodud välja TFT LCD ekraan koos ühendustega nii eest kui tagant vaates. Tabelis 2.3 saab tutvuda ühenduste kirjelduste ja Atmega328 mikrokontrolleri vastavate ühendus portide viidetega.



Joonis 2.7: Arduino Graafilise TFT LCD eest vaade koos ühendus viikudega



Joonis 2.8: Arduino Graafilise TFT LCD ekraani tagant vaade

Tabel 2.3: LCD-mooduli ühendus viigud

Ühendus-lüli	**Sümbol**	**Kirjeldus**	**Atmega328 port**
1	+5V	Toitepinge (3,3V loogika kasutatakse ka vastavat pinget)	+5V
2	MISO	Ekraan saada andmeid Arduinole	D_{12}
3	SCK	Andmete sünkroniseerimise taktsignaal	D_{13}
4	MOSI	Arduino saadab andmeid Ekraanile	D_{11}
5	LCD CS	LCD ekraani valiku signaal	D_{10}
6	SD CS	Mälukaardi valiku signaal	-
7	D/C	Andmete / Käsustiku režiimi valik	D_9
8	Reset	Ekraani algseadistamise signaal	D_8
9	BL	Tagataustavalguse toitepinge +5V (tugevus reguleeritav PWMga)	+5V või PWM
10	Ground	Ühendus maaga	GND

2.2.4. UV-A LED

UV-A LEDid on antud seadme kõige tähtsam osa. LEDid peavad vastama kindlale spektrile, mis kivistavad akrüüle vastavalt nende keemilistele seaduspärasustele. Erinevate tootjate andmetele tuginedes valmistatakse LEDile mõeldud UV geellakid kivistuma enamasti elektromagnet kiirgusspektri vahemikus 370-380 nanomeetrit.

Sellest infost lähtudes on valitud antud 3w UV-A LEDid 370-380nm. Kogu akrüüle kivistava spektri katmiseks vajalikud LEDid läheksid prototüüpmudeli koostamiseks põhjendamatult kalliks. Siiski on tulevikus prototüüpi arendades võimalus lisada spektri laiendamiseks ka madalama kiirgusega LEDe. Tabelis 2.4 on valitud LEDide tähtsamad elektri-optilised omadused[29].

Tabel 2.4: UV-A 370-380nm LEDi põhiomadused

Elektri-Optilised omadused I=750mA, T=25C juures				
Parameeter	**Sümbol**	**Miinimum**	**Tüüpiline**	**Maksimum**
Kiirgus võimsus	θe	900	~	1000mW
Lainepikkus	λD	370	~	380nm
Päripinge	V_F	3.5	~	45V
Võimsustarve	P_D	2,63	~	3,38W
Kiirgus nurk	2θ1/2	~	120	~
Soojustakistus	$R\theta_{J-B}$	~	12°C/W	~
Pärivool	I_F	-	~	750mA
Ühenduslüli temperatuur	Tj			115°C
Opereerimise temperatuur	T_{opr}	-40°C	~	60°C
LED mõõted	L*P*K	14.5 x 8,05 x 5,1mm		
Radiaatori mõõtmed	L*P*K	19,8*19,8*1,6mm		

2.2.5. Ventilaator

Kuna suure võimsusega LEDid toodavad palju kuumust, siis õhuvahetuseks on valitud 40x40x10mm ventilaator Sunon MagLev ME40101V1-000U-G99[32]. Eelistused

puudusid, kuid määravaks said toite pinge 12V, mõõtmed ja müratase. Antud tootevaliku omadusteks on harjadeta mootor, ventilaatori pöörete signaali lugemise ühendus viik.

Ventilaator töötab väljatõmbe režiimis. Kuna mootorit ei saa tagurpidi tööle panna, siis tuli keerata antud labad nii, et see tõmbaks korpusest kuuma õhku välja tekitades nii ala rõhuga külmale õhule kiirema juurdepääsu LEDide radiaatori efektiivsemaks jahtumiseks. Tabelis 2.5 tähtsamad elektrilised karakteristikud.

Tabel 2.5: Ventilaatori elektrilised karakteristikud

Sunon MagLev ME40101V1-000U-G99 omadused	
Mootori disain	2 faasiline, 4-mähisega harjadeta DC
Tööpinge	12VDC
Töö vool	90mA / Max. 104MA
Võimsustarve	4.5 ~ 13.8VDC
Miinimum käivitus pinge	4.5VDC (25C)
Töö temperatuur	-10C kuni +70C
Hoiustamise temperatuur	-40C kuni 70C

2.2.6. Toiteadapter

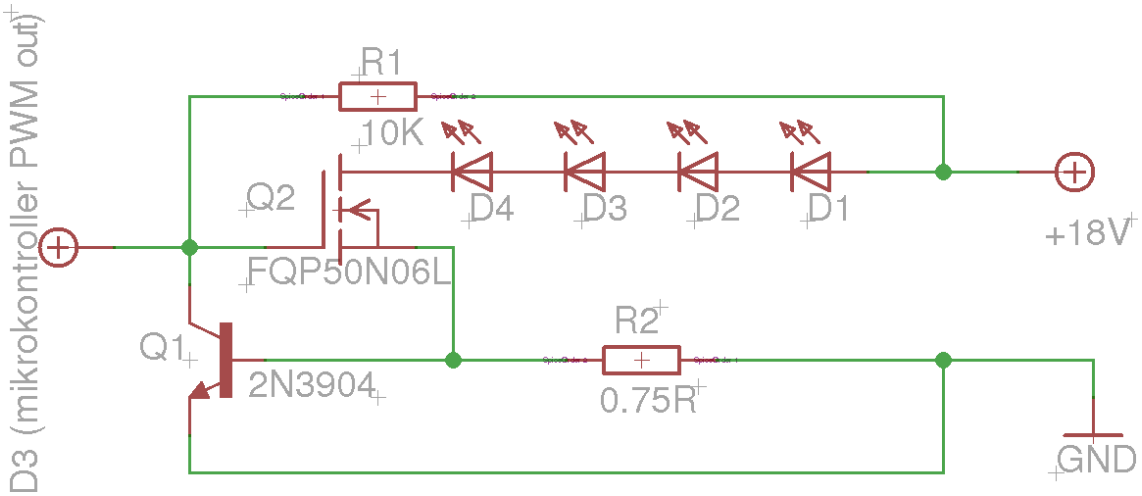
Toiteadapteriks valisin kodus leiduva 16V 4.5A sülearvuti laadija Lenovo AC/DC Combo Adapter P/N 41R0140[47]. Kuna 16V toide pakub piisavalt pinget 4 korda 3.5V – 4.5V LEDide järjestik ühendamiseks ning voolu tarve jääb 700mA juurde, siis sai valik tehtud ühehäälselt selle adapteri kasuks. Valida oli veel 12V 5A ja 12V 2A toite adapter, kuid toitepinge LEDide järjestik ühendamiseks jäi mõlemal toiteplokil piiripealseks ning 2A toiteplokk ka nõrgaks seerias ühendamise puhul. Ülejäänud moodulid nagu ventilaator ja mikrokontroller koos arendusplaadiga tarbivad tühisel määral voolu, mis valiku puhul takistavaks teguriks ei saanud.

2.2.7. Reaalne prototüüpmudel

Prototüüpmudeli ehitamiseks kasutatakse Iteaduino UNO arendusplaati. Arendusplaadile ühenduste laiendamiseks on appi võetud maketeerimislaud; ühenduste

tegemiseks maketeerimislaua juhtmed. Mõned komponendid on joodetud ka makettplaadi külge ning teiste moodulitega ühendamiseks on kasutatud cat5 kaabli kiude nagu näiteks IR-andur, juhtnupud. UV-A LEDid tarbivad suhteliselt palju voolu, seega nende toite ahela kindlustamiseks on erandina võetud 0.75mm^2 läbilõikega vaskkiud juhtmed. Korpus mudelil on prototüübi arendamise ajaks vahtplastist, mis ühendab kõiki elektroonika komponente ühtsesse ruumi. Illustreerivat pilti reaalsest prototüübist vaata Lisast 3.

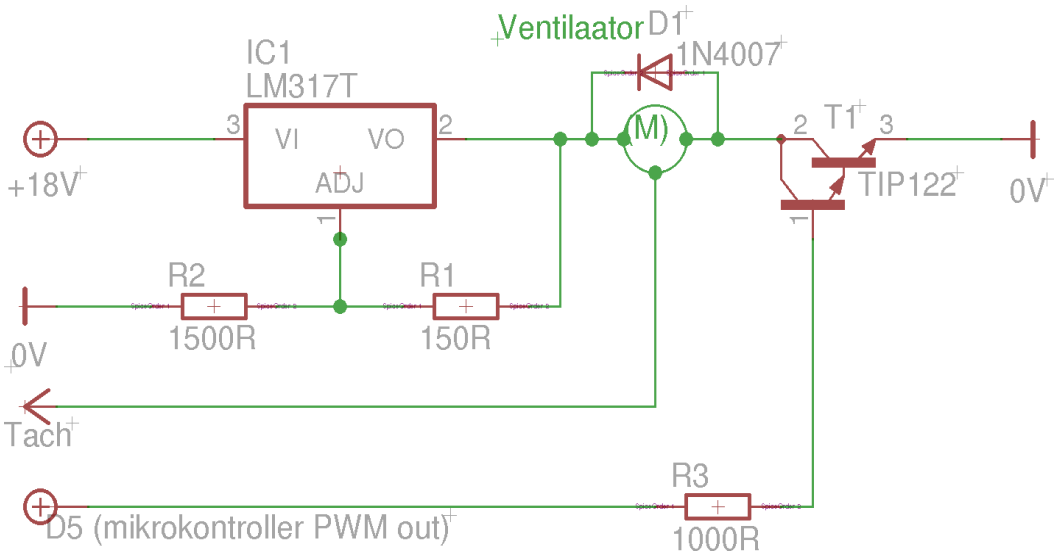
UV-A LEDe juhitakse mikrokontrollerist PWM signaaliga. Toite loogika skeem[50] koosneb N-kanali QFET MOSFETist (FQP50N06L)[48], bipolaarsest npn transistorist (2N3904)[49] ja kahest takistusahelast skeemis. MOSFET Q2 töötab kui muutuva takistusega takisti, mida lülitab sisse ja välja R1 takistiga ühenduses olev mikrokontrolleri ühenduslüli. LEDe toitev vool käib läbi Q2e ja R2 takisti. Kui R2 takistit läbib liiga palju voolu, siis hakkab npn transistor Q1 voolu MOSFETil Q2 piirama. R2 takisti arvutatakse välja põhimõttel, et kui palju voolu me tahame läbi tarbija lasta. Pinge muut npn transistori jalgadel V_{EB}=0.5V ja teame, et LEDid tarbivad 700mA voolu. Seega $R_3 = \frac{V_{EB}}{I_{LED}} = \frac{0.5\,V}{0.7\,A} = 0.71\,\Omega$, mille lähim vaste on 0.75Ω ja miinimum tarbimisvõimsuseks saame $P = U * I = V_{EB} * I_{LED} = 0.5 * 0.7 = 0.35\,W$. Seega tuleb valida takisti 0.75Ω ja minimaalselt 1/2W. Joonisel 2.9 LED PWM lülituse skeem välja toodud.



Joonis 2.9: UV-A LED lülitus reguleerimise skeem

Suure võimsusega LEDe on vaja alati jahutada. Kuigi tarbijal on endal tehase poolt kaasas juba passiiv jahutus – milleks on alumiinium plaat tagaküljel, aga prototüübis plaanitakse kasutada rohkem kui ühte LEDi siis ühendatakse kõik LEDid ühe ja suurema passiiv jahutus radiaatori külge. Siiski alati ei suudeta sellega optimaalseimat temperatuuri tagada. Siin tuleb mängu ventilaator, mis on mõeldud passiiv jahutusele väljastpoolt tuleva jaheda õhuvoolu kiiremaks vahetamiseks. Kuna antud ventilaator nõuab omaduste poolest maksimaalselt 13.8V pinget töötamiseks ja seadme kiirust plaanime juhtida jällegi PWM meetodi abil, siis tuleb tema ühendamiseks vooluahelasse projekteerida jällegi lülitus skeem[52]-[53]. Prototüüpmudeli toiteadapter võib olla suurem kui 12V ja ventilaatorile maksimaalse lubatud pinge tagamiseks vajame siinkohal pinge regulaatorit näiteks LM317T[51] (Joonisel 2.10) IC1. Takistiga R1 ja R2ga saame väljund pinge V_O meile sobivaks valemiga:

$$V_O = V_{REF}\left(1+\frac{R2}{R1}\right) \quad , \quad V_O = 1.25\,V\left(1+\frac{1500\,\Omega}{150\,\Omega}\right) = 13.75\,V \quad .[53]$$

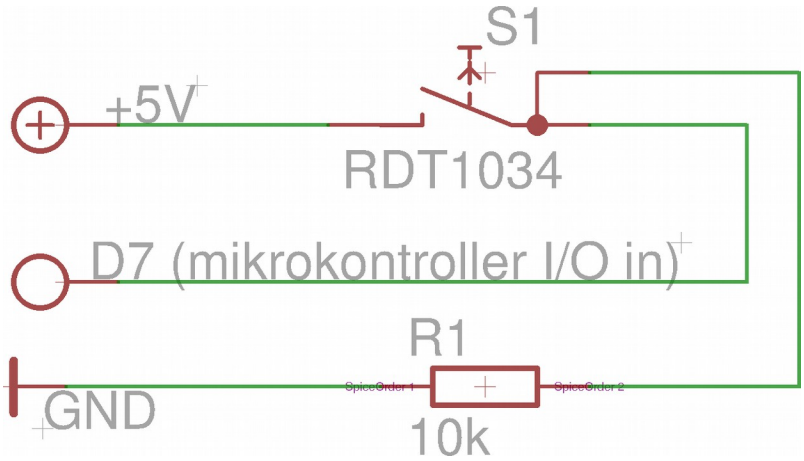


Joonis 2.10: Ventilaatori pinge regulaator ja PWM lülitus skeem

Vahemärkusena on joonisel 2.10 välja toodud „tach" ühenduslüli, millega vajaduse korral saab mikrokontrolleri digitaalsisendiga lugeda ventilaatori tagasiside impulssi. See info võimaldab meil välja arvutada ventilaatori kiiruse. Antud prototüüpmudelit ehitades pole see, aga kuigi oluline ning keskendume me PWM signaali saatmisega mikrokontrollerist. Sarnaselt LEDi tugevuse reguleerimisele juhime ka ventilaatorit, aga selle vahega, et siin on katsetamise mõttes kasutatud Darlingtoni tüüpi transistori TIP122[54] (Joonisel 2.10) T1. Kuigi TIP122 andmete lehe järgi sisaldab antud

transistor ka dioodi, siis paneme ka ventilaatori otstele igaks juhtumiks täiendava dioodi vältimaks skeemi läbipõlemist.

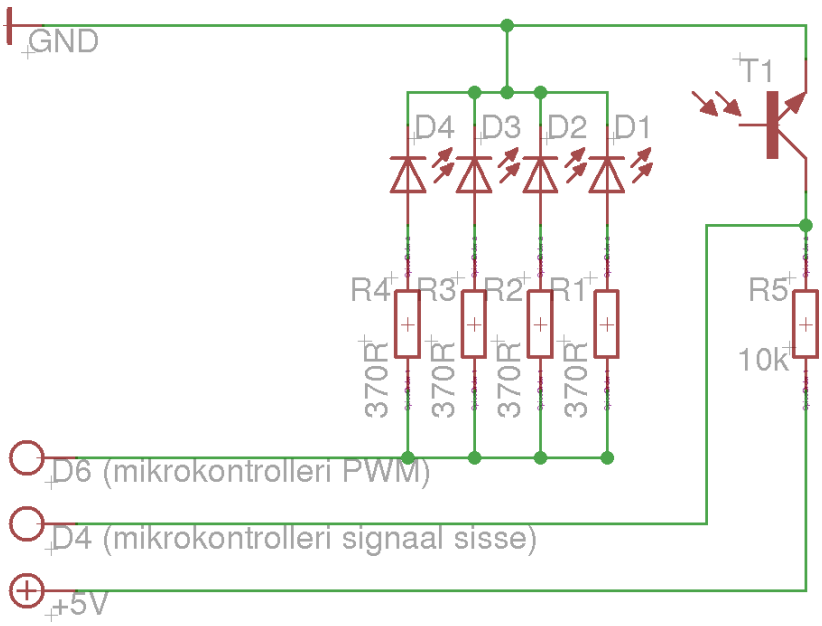
Järgmine moodul on juhtnupp taimeri aja valimiseks (vaata joonis 2.11). Skeem[55] on üsna lihtne, sest koosneb nupust, takistist ja omavahelistest ühendustest. Saame nn „on-off" lüliti, mis alla vajutatuna annab mikrokontrollerile signaali +5V ehk lülita. Lahtises olekus takisti R1 mikrokontrolleri digitaalsisendi ja maa vahel tõmmatakse pinge ahelas 0V, mis tähendab ära lülita või lülita välja.



Joonis 2.11: Mikrokontrolleri juhtnupu skeem

Neljandaks tähtsaks skeemiks[56] prototüüpmudeli puhul on IR-lähedus andur (vaata joonis 2.12). IR anduri ülesandeks on tuvastada käsi UV-LED lampide all, et need siis koos taimeriga sisse lülitada. D_1-D_4 on IR dioodid[57], mis kiirgavad infrapuna valgust valitud sagedusel mikrokontrolleri ühenduses D6 ehk mikrokontrolleri digitaalne väljund signaal. T_1 on fototransistor[58] või fotodiood, mis on tundlik infrapuna valgusele nii, et see muudab vastavalt intensiivsusele enda sisetakistust andes signaali D_4 ehk mikrokontrolleri analoog sisendile. IR dioodi D_1-D_4 takistuse arvutame välja põhimõttel, et mikrokontrolleri väljund vool on maksimaalselt 0,04A. Tahame kasutada kindluse mõttes 4 IR dioodi signaal võimalikult suureks hajutamiseks, seega jagame kontrolleri väljund voolu neljaks ja saame 0,01A dioodi kohta. Andmelehelt[57] saame teada, et normaal pinge on 1,3V anoodi ja katoodi vahel, seega kuna kontrolleri väljund pinge on 5V, siis väärtuse takisti saame:

$$R_{1,2,3,4} = \frac{(5\,V - 1{,}3\,V)}{0{,}01} = 370\,\Omega$$

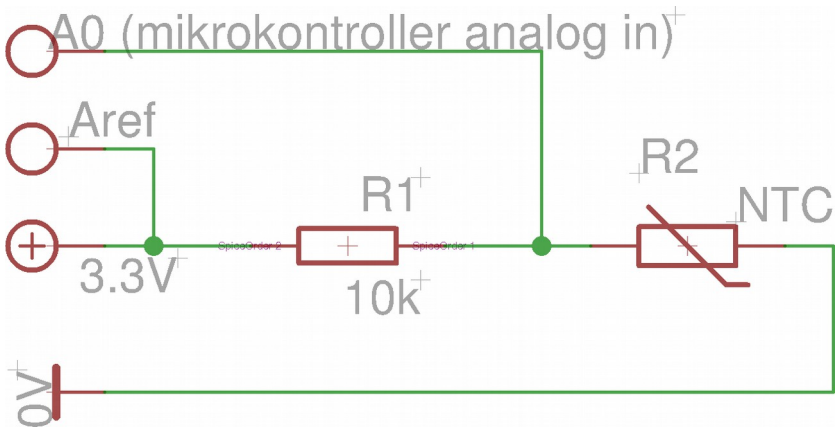


Joonis 2.12: IR-lähedus anduri skeem

Selleks et hinnata õigesti passiivjahutuse temperatuuri on asjakohane paigaldada radiaatori külge NTC *thermistor[59]*, mis on tegelikult termotakisti (Vaata skeemi[60] 2.13). Tema takistus 25C kraadi juures on 10kΩ. Temperatuuri suurenedes takistus väheneb ja vastupidi. Kuigi antud seadme juures pole temperatuuri täpne mõõtmine eriti oluline ja välja on jäetud ka termotakisti kalibreerimine, on antud skeemis katsetamiseks kasutatud siiski 3,3V loogikat. See tähendab, et 10-bitise A/D konverteri[43] korral

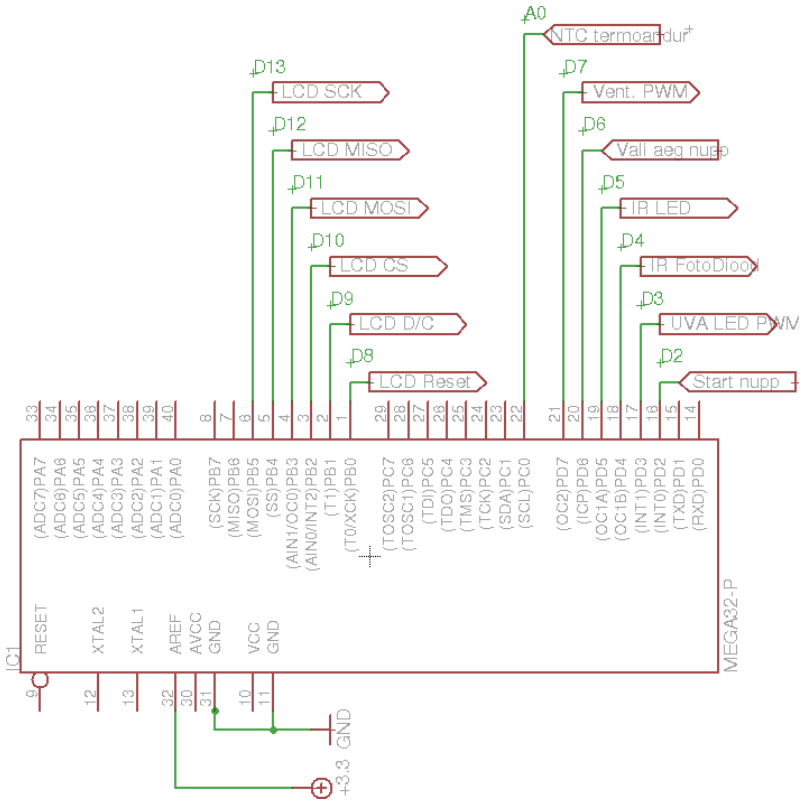
$$U = \frac{5V}{1024_{ühikut}} = 0{,}00488\,V \quad \text{sammu} \quad \text{asemel} \quad \text{saame} \quad U = \frac{3{,}3V}{1024_{ühikut}} = 0{,}00322\,V$$

temperatuuri ühe ühiku mõõtmise vahemikuks, mis peaks olema mõnevõrra täpsem. Teiste sõnadega mikrokontrolleri sisendiga A0 võtame 10-bitise täpsusega NTC termotakistilt lugemi millivoltides, mis konverteeritakse 0-1023 ühikuks. Teades, et takistus on 25C juures ~10kΩ arvutame tarkvaraliselt välja ligikaudse seadme passiivjahutuse temperatuuri.



Joonis 2.13: NTC termoanduri skeem

Kõik eelpool kirjeldatud moodulid ühenduvad mõistagi otseselt Atmega328 mikrokontrolleri külge (vaata Joonist 2.14). Olenevalt moodulist saadetakse signaali või võetakse seda mikrokontrolleri portidest vastu. Legendis märgitakse sisend signaalid noolsiltidena teravikuga mikrokontrolleri suunas ja väljund signaalid teravikuga mikrokontrollerist vastassuunaga. Analoog signaali konverteerimine digitaalseks sisendiks on märgitud prefiksiga A ning digitaalne sisend-väljund prefiksiga D, mille lõpus on number moodulite identifitseerimiseks.



Joonis 2.14: Atmega328 moodulite ühenduste legend

3. Atmel Atmega328 mikrokontrolleri programmeerimine

Järgneva paragrahvi eesmärk on anda ülevaade prototüübi Arduino IDE keskkonnast ning prototüüpmudelile loodud tarkvarast antud platvormi peal. Tutvustada lühidalt kõikide moodulite tarkvaralist funktsionaalsust.

3.1. Arduino IDE olemus ja tööprotsess antud platvormil

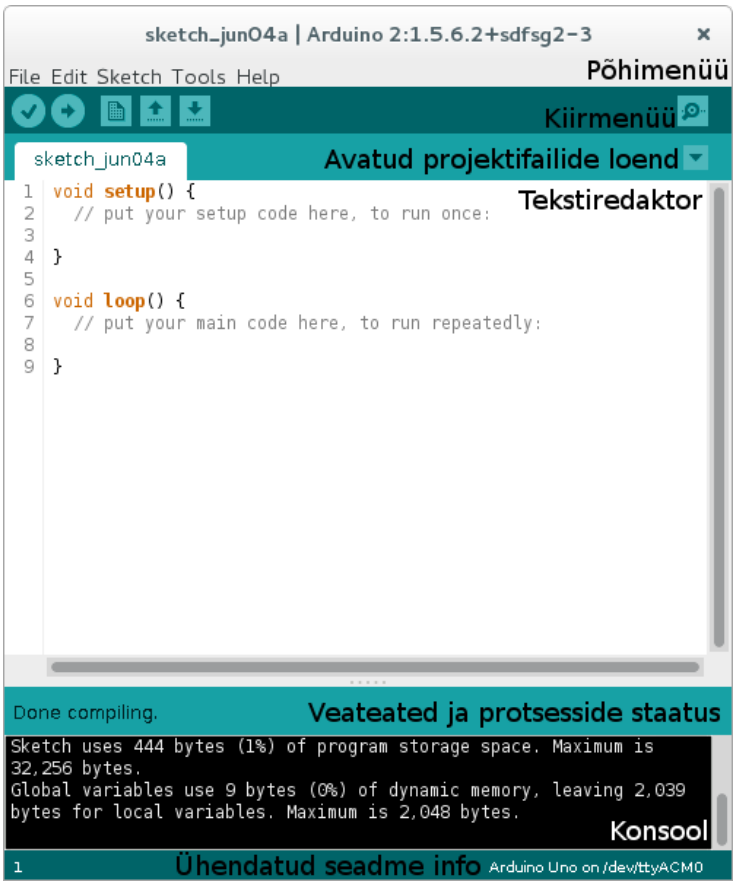
Arduino IDE platvorm on integreeritud moodulite kogum alates programmkoodi kirjutamisest, kuni selle mikrokontrollerisse laadimiseni enamasti läbi USB pordi. Antud arendusplatvorm saab kenasti hakkama kirjavigade leidmise ja nende raporteerimisega konsooli. Saab aru süntaksi hälbest ja lihtsustab koodi kirjutamist nii eri värviliste lause osade kui ka kirjavahemärkide automaat esile toomisega, mille eesmärgiks on koodi selgem ja kiirem kirjutamine ilma vigu tegemata.

Platvorm suudab ühe nupu vajutusega eel-verifitseerida koodi enne kompileerimist ja mikrokontrollerisse ülesse laadimist. Oskab suhelda erinevate Arduino arendusplaatidega läbi USB liidese ning tarkvara transleerida masinkoodiks erinevatele Atmel tüüpi mikroprotsessoritele suurema vaevata. Arendamise kiirendamiseks ja lihtsustamiseks tulevad installeerimisel kaasa erinevad tööriistad ja rikkalikud teegid mitmekesiste riistvara moodulitega suhtlemiseks koos näite materjaliga.

Sisse ehitatud tööriistadest väärivad mainimist kirjutatud koodi automaatne formaatmine ning teksti kodeeringu parandamine. Arhiveerida salvestab toodetud koodi pakitud faili kettale. Keskkond pakub võimalust vahetada manuaalselt arendusplaati ja suhtlus porti sama tarkvara erinevatele platvormidele kompileerimiseks. Järjestik ühenduse loomisega saab monitorida juba mikrokontrollerisse laetud töötavat koodi selle silumise eesmärgil. Üheks suureks eeliseks on koodi kirjutamise võimalus samaaegselt avatud laiali paisutatud failidesse parema ülevaate saamise saamiseks. Arenduskeskkond on loodud nii, et see eeldab USB olemasolu arendusplaadil. Siiski mikrokontroller peab omama eelinstalleeritud käivitusprogrammi (*bootloader*it) kiireks ja mugavaks USB ühenduseks. Kui viimast pole, on võimalik see Arduino IDE

keskkonnas mikrokontrollerile paigaldada. Samuti on võimalik programmeerida ka eraldiseisvat mikrokontrollerit.

Sellise integreeritud keskkonna abil ei pea ise projekti loomiseks suurt midagi tegema. Uus projekt avatakse automaatselt ning kohustuslikud programmkoodi meetodid on juba eel täidetud uue faili loomisel. Kõik erinevad funktsionaalsused on ühte liidesesse integreeritud (vaata Joonis 3.1). Kiirmenüü alla on toodud vasakult paremal lugedes nupud: silu koodi, lae kontrollerisse ning uus fail, ava fail, salvesta fail ning paremal pool ääres „Serial-monitor" koodi reaalajas silumiseks. Alla poole liikudes avaneb liidese keskosas tekstiredaktor, mille koodi read on nummerdatud ja programmkood värvitud. Järgmise rea moodustavad veakirjeldused ja liideses toimuvate protsesside staatuse teated. Konsool – valge tekst mustal taustal näitab programmkoodi kompileerimise veateateid ning silumise informatsiooni. Kõige viimane rida näitab infot ühendatud arendusplatvormist.



Joonis 3.1: Arduino IDE avatud kasutajaliides

3.2. Mikrokontrolleri minimaalne käivitus kood.

Mikrokontrolleri minimaalseks käivitamiseks Arduino baasil mikrokontrolleris vajame kahte meetodit:

```
void setup() {  // Siia kood, mida käivitatakse mikrokontrolleri käivitamisel }

void loop() {  // Siia kood, mida korratakse tsükliliselt kuni mikrokontolleri vooluvõrgust lahti ühendamiseni. }
```

Need meetodid ei sisalda, midagi ja ühetgi funktsionaalselt omadust neil ei ole, kuid ilma selleta pole võimalik mikrokontollerit ka käivitada ja muid funktsioone lisada.

3.3. Mikrokontrolleri ja TFT vedelkristall kuvari vaheline kommunikatsioon

Antud moodul on prototüüpmudelis ainuke seade, mis kuvab visuaalset informatsiooni. Tegemist on Arduino poolt toetatava välise seadmega. Kõik vajalikud teegid programmeerimiseks ja LCD edukaks kasutamiseks on juba eel-defineeritud, mis kätte saadavad ka avatud lähtekoodina[61]-[62]. Need teegid omakorda baseeruvad Arduino SPI teekidel[34] ehk sünkroonsel järjestikandmevahetuse protokollil. Selletõttu antud töös ei süvene autor nendesse teekidesse sügavuti vaid kasutab eeldefineeritud meetodeid ja lähtekoodi.

Niisiis vajalikud teegid LCDga suhtlemiseks on:

```
#include <TFT.h> //Arduino LCD library

#include <SPI.h> //Serial Peripheral Interface communication library
```

TFT.h teegi sees luuakse uus klass nimega TFT, mis toetub baas klassile Adafruit_ST7735.h. TFT klassi sees pannakse käima meetod, mis kutsub välja kõik vajalikud funktsioonid LCD ekraani kontrolleri initsialiseerimiseks. Peale seda on alles võimalik järgmisi toiminguid edasi teha.

```
/// The Arduino LCD is a ST7735-based device.
/// By default, it is mounted horizontally.
/// TFT class follows the convention of other
/// Arduino library classes by adding a begin() method
/// to be called in the setup() routine.
```

```
class TFT : public Adafruit_ST7735 {
public: TFT(uint8_t CS, uint8_t RS, uint8_t RST);
void begin();
};
```

Rohkem sügavamale pole autoril teekidesse programmkoodi kirjutamiseks esialgu vaja enam süüvida. Peame veel niipalju teadma, et alates sellest hetkest on hõivatud ekraani kasutamiseks MOSI, MISO, SCK jaoks vastavalt 11, 12, 13 pordid mikrokontrolleril. Ise peame defineerima CS, DC ja RST. Autor teeb seda vastavalt soovitusele juhendist[33] ehk vastavalt mikrokontrolleri pordid 10, 9, 8. Autori koodis näeb see välja nii:

```
/* mis mikrokontrolleri pordid on kasutusel */
#define cs 10 // Defineeri CS, pordi nr D10
#define dc 9 // Defineeri CS, pordi nr D19
#define rst 8 // Defineeri CS, pordi nr D8
TFT TFTscreen = TFT(cs, dc, rst); //Loo TFT ekraani instants.
```

Järgmiseks defineeritakse autori poolt TFT.h klassi instants ja kirjutatakse mikrokontrolleri *void setup()* meetodi sisse klassi initsialiseerimise käsk, et oleks võimalik LCDga teha vajalike toiminguid ehk kuvada infot. Kuna klassi kasutamine on üpris triviaalne, siis peale all pool oleva instantsi loomise ja „teretulemast" teksti kuvamise ning info kustutamise näite on täielik kood ülevaatlikult olemas Lisas 2.

void setup(){} meetodi sees olev ühekordselt käivitatav kood:

```
TFTscreen.begin(); //Initsialiseeri TFT LCD ekraan
TFTscreen.background(0,0,0); //Sea tagataust mustaks
TFTscreen.stroke(255,255,255); //Sea teksti värv valgeks.
TFTscreen.setTextSize(2);   //Sea teksti suurus 2
TFTscreen.text("T e r e t u l e m a s t !", 3, 55); //Kirjuta tekst „T e r e t u l e m a s t !„ vasakult ekraani nurgast kolm pikslit paremale poole ja ülevalt nurgast 55 pikslit alla poole
delay(3000); oota 3000ms ehk 3sekundit.
TFTscreen.stroke(255,255,255); //Sea teksti värv samaks tagatausta värviga
 TFTscreen.text("T e r e t u l e m a s t !", 3, 55); //Kirjuta tekst „T e r e t u l e m a s t !„ tagatausta värviga üle, ehk kustuta tekst ekraanilt.
```

3.4. Seadme kasutaja liidese menüü esialgne realiseerimine

Seadme sisse lülitamisel kuvatakse peale tervitusteksti lõppemist kasutaja liidese menüü. Menüüst saab kasutaja „Vali aeg” nupu abil valida endale sobiva lambi töötamise aja geellaki polümeriseerimise läbi viimiseks. Vaata meetodi nägemiseks Lisa 2 sektsiooni *[Lambi menüü meetod]*.

Ekraanile kuvatakse kiri „Select time:” ehk vali aeg. Samal ajal käivitub do while() tsükkel.

```
TFTscreen.text("Select time:", 0, 0);
do {
        //.....
        if (digitalRead(IRsensorPin) == LOW)
        { startTimerSignalStatus = HIGH; }
        //.....
        if (selectButtonStatus == HIGH)
        { //kuva järgmine aeg massiivist }
        //.....
}
  while (startTimerSignalStatus == LOW);
```

Tsükkel hakkab kontrollima, kas kasutaja on valinud töörežiimi pikkuseks uue väärtuse: Võimalikud lambi LEDide sisselülitamise pikkused on: 30, 60, 90, 120, 180 sekundit. Kui uus väärtus on valitud, aga IR-lähedus andurilt pole veel signaali saadud , siis antakse kasutajale uus võimalus valida töörežiimiks uus aeg. Seda seni kaua kui kasutaja on asetanud käe UV LEDide alla, siis suundub seade töörežiimi. Peale LEDide kustumist kuvatakse kasutajale uuesti menüü ja valitud aeg, et protseduuri korrata, valida uus aeg menüüst või seade välja lülitada.

3.5. Lambi töörežiim ehk taimeri meetodeid

Kui seade on saanud positiivse signaali startTimerSignalStatus väljale, lülitatakse LEDid sisse PWM meetodiga maksimum võimsusele. Antud juhul on ÜWM sagedus skaala 0-255 ühikut 8-bitises süsteemis. Autor on teinud selle jaoks meetodi:

```
turnLED(255);
```

Järgneb tsükkel, mille iga ringi täitumisega loetakse menüüst valitud ajast 1 sekund maha. Selle sekundi sees kontrollitakse seadme LEDide temperatuuri meetodiga:

```
getTemperature();
```

Temperatuuri kuvatakse prototüübil testimise mõttes ka üleval vasakus nurgas pisikeses kirjas. Väärtus salvestatakse globaalsesse muutujasse teistele meetoditele kättesaadavasse muutujasse

```
float Temperature;
```

Edasi võetakse viimasest muutujast temperatuuri võrdlus lubatud maksimum temperatuuriga ja vastavalt võrdluse tulemustele lülitatakse ventilaator sisse või mitte. Meetodiks kasutame

```
turnFAN(true); //Lülita ventilaator sisse
turnFAN(false);//Lülita ventilaator välja
```

Järgmisena oodatakse stopTimerSignal sisendist nupuvajutust. Kui kasutaja on vajutanud pikalt (vähemalt 1s) nuppu all, siis signaal on positiivne ja väljutakse töörežiimist. Kui nuppu pole vajutatud jätkab taimer oma tööd. Peale taimeri loendamise nulli jõudmist lülitatakse LEDid välja PWM sagedus lülitatakse madalaimale tasemele 0.

```
turnLED(0);
coolingFan();
```

Kontrollitakse üle ka temperatuur ja kui ventilaator juba töötab, siis peale töörežiimist väljumist jäetakse ventilaator veel kümneks lisasekundiks tööle. Sel juhul tuleb ekraanile tekst „Cooling...” - ehk jahutan. Vastasel juhul tuleb ekraanile tekst viieks sekundiks „Finished!”, mis tähendab, et loendur jõudis nulli ja tsükkel sai läbi. Lisas 2 sektsioonis *[Lambi töörežiim ehk taimeri meetod]* on meetodit võimalik lähemalt uurida.

3.6. UV-A LEDide juhtimine

UV LED dioodide juhtimine on toodud välja eraldi meetodisse, et antud koodi mitte dubleerida nii aja kui ruumi kokkuhoiu mõttes. Järgnevalt meetodi kirjeldus, kuid sellega on võimalik tutvuda Lisa 2 sektsioonis *[UV-A LED juhtimine]*.

```
void turnLED(int brightness)
{
 if (brightness == 0) //Kui tahame ledi LED välja lülitada
        analogWrite(ledPWM, brightness); //Lülita LED
välja andes ette PWM sageduse 0
else //Kui tahame LEDi sisse lülitada
   for (int counter = 0; counter <= brightness;
counter++) //Lülita LED tsükliliselt sujuvalt sagedust
tõstes sisse kuni maksimum ette antud heleduseni
        {       analogWrite(ledPWM, counter);
                delay(5); //5ms intervalliga
        }
}
```

3.7. Ventilaatori kontroll

Ventilaatori juhtimise meetodi täielik lähtekood on toodud Lisa 2 sektsioonis *[Ventilaatori juhtimine]*. Ventilaatori meetod on nimega

```
void turnFAN(bool turnOn){}
```

Ventilaatori juhtimine käib sarnaselt LEDidele ehk analogWrite() meetodiga, mille sagedusvahemik saab olla 0-255. Siiski harjadeta ventilaatorit on kehv PWMiga juhtida kui ei ole PWM väljundit (väljund juhet). Et pöördeid 5V pulseerimispingega mõjutada tuleks ehitada ventilaatorile vahele PWMiga digitaalselt reguleeritav pinge jagur. Üks võimalus on Kasutada LM317T regulaatorit ühendades teda operatsiooni võimendiga ja madalpääs filtriga. Alles see järel saaksime digitaalselt kontrollitava pinge jaguri täies ulatuses ventilaatori pöörete reguleerimiseks. Antud prototüübi juures sobib meile ka väheste pöörete reguleerimine ja isegi maksimum pööretel ventilaatori käitamine, kuna LEDidega on digitaalne PWM signaal antud töös piisavalt tõestanud. Prototüübi edasi arendamise korral kaalub autor lisada digitaalselt juhitava pinge jaguri ventilaatori pöörlemiskiiruse reguleerimiseks.

3.8. IR-lähedus anduri loogika

IR-lähedus anduri tarkvara koosneb kahest komponendist nagu ka riistvaraline pool (Vaata peatükk 2 Joonis 2.12), kus on toodud IR LED ja IR fotodiood. Niisamuti tarkvaras peame saatma IR LEDiga signaali, mille peegeldumist kehalt ootame

fotodioodile. LED fotodiood soovib ~38kHz sagedust saatjalt. IR LED meetod on järgmine:

```
void SendIRSignal()
{
  for (int i = 0; i <= 384; i++) { //Korda järgmist tsüklit
384 korda
    digitalWrite(IRledPin, HIGH); //Lülita IR LED sisse
    delayMicroseconds(13); //Oota 13ms
    digitalWrite(IRledPin, LOW); //Lülita IR LED välja
    delayMicroseconds(13); //Oota 13ms
  }
```

Meetodiga[63] kindlustame, et LED plingiks 38kHz ligilähedasel sagedusega. Arvutus on järgmine, me tahame, et LED lülituks välja ~38000 korda sekundis.

$\frac{1\,s}{38000} = 0.000026\,sekundit \Rightarrow 26\,mikrosekundit$ lülitame LEDi sisse ja välja ühe poole ajast ja sisse teise poole ajast. Ning saame 13ms sisse lülitatu ja 13ms välja lülitatud LEDi.

Fotodioodina kasutab autor antud töös fototransistori fotodioodi signaali vastu võtmise tarkvaralise lahenduse leiab Lisa 2 sektsioonis *[Lambi menüü meetod]*.

```
SendIRSignal();//Saadame ~38kHz signaali.
    if (digitalRead(IRsensorPin) == LOW)
    {
//Kui IR sensori signaal on madal, siis tähendab see
seda, et kehalt peegeldatakse fotodioodi ehk IR sensor
ahelasse IR LEDilt saadud signaali
    }
    else {
//IRSensor signaal on kõrge, kehalt peegeldust ei ole
}
```

3.9. Termoandur

Termoanduri meetod täielikus lähtekoodis on Lisa 2 sektsioon *[Temperatuuri lugemine]*. Lähte koodist on väga lihtsalt võimalik aru saada valemitest kuidas termotakistilt takistuse järgi temperatuuri arvutatakse, aga idee[41]-[42], [60] on lihtsalt

selles, et võtame 3.3V referents pingega mikrokontrolleri analoog näidu A0 pordist 10-biti skaalas ehk jaotuseks saame 0-1023 ühikut, mille sammu vahe on 0.0032V. Võtame viis näitu 10ms vahedega ning arvutame sellest mediaan keskmise. Teisendame analoog sisendilt saadud viie tulemuse mediaan keskmise 1023 punkti skaalal ning jagame skeemis vastukaaluks pandud takisti väärtuse mediaaniga läbi. Mediaani enda jagame läbi termotakisi enda nominaal takistusega eeldusel, et ideaal temperatuur on 25C 10 kiloomi juures nii nagu NTC andmelehel[59] kirjas. Võtame vastusest logaritmi pöördväärtuse nominaal temperatuurist liita beeta koefitsiendiga liidame temperatuurile mille me just beetaga läbi jagasime ning võtame kogu saadud summast uuest pöörd väärtuse ja lahutame kelvinid ning saame temperatuuri Celsiuse skaalas. Tarkvaraline loogika järgnevalt välja toodud

```
 float TempMedian; //Temperatuuri mediaan keskmise muutuja
 /* Võtame  5 näitu 10ms vahedega A0 pordilt*/
 for (i = 0; i < NTC_NR_OF_SAMPLES; i++) {
  NTCSampleArray[i] = analogRead(NTC_PIN);
  delay(10);
 }

 /* Liidame näidud*/
 TempMedian = 0;
 for (i = 0; i < NTC_NR_OF_SAMPLES; i++) {
  TempMedian += NTCSampleArray[i];
 }
 /*Arvutame keskmise*/
 TempMedian /= NTC_NR_OF_SAMPLES;
 TempMedian = 1023 / TempMedian - 1;
 TempMedian = NTC_RESISTOR / TempMedian;

 /*Arvuta ja konverteeri temperatuur Celsiusesse*/
 Temperature = TempMedian / NTC_RESISTANCE;// (R/Ro)
 Temperature = log(Temperature); // ln(R/Ro)
 Temperature /= NTC_BETA_COEFFICIENT; // 1/B * ln(R/Ro)
```

```
  Temperature += 1.0 / (NTC_NOMINAL_TEMP + 273.15);
// + (1/To)

  Temperature = 1.0 / Temperature;// Inverteerime
väärtuse

  Temperature -= 273.15;// Konverteerime kelvinist
Celsiusesse
```

4. Kokkuvõte

Käesoleva bakalaureuse töö eesmärgiks on teha ülevaade füüsiliselt realiseeritud UV-A geel LED lambi prototüüpmudelist. Tutvustatakse mikrokontrollerit ja selle hallatavaid mooduleid ning arendatud tarkvara koos tarkvaraarendus platvormiga. Tehakse ülevaade lõplikust prototüüpmudelist tervikuna.

Prototüübi südames on Atmel Atmega328 mikrokontroller. Töö ülesandeks seatakse tarkvara arendamine antud mikrokontrollerile vajaliku funktsionaalsuse saavutamiseks. Vajaliku ülesande täitmiseks on vaja realiseerida ka teatud moodulite riistvaralised lahendused. Kuna seade on peale lõpliku valmimist mõeldud üsna laiale kasutajaskonnale, siis intuitiivseks ja atraktiivseks kasutamiseks on seadmele lisatud värviline LCD ekraan. Prototüübi valmistamise arendusplatvormiks on valitud Iteaduino UNO (Arduino UNO kloon) – antud platvorm on suhteliselt odav ja katab prototüübi arendamiseks vajalikud tingimused mõningate mööndustega. Autor valis antud seadme suurest huvist Arduino platvormi ja automaatika vastu. Samas saab seda arendusplaati kasutatada tulevikus ka teiste projektide arendamise tarbeks.

Töö koosneb praktilisest ja kirjalikust osast. Praktiline osa jaguneb omakorda kaheks, millest üks moodustab seadme ehitamise ning moodulskeemide väljatöötamise ning teine osa hõlmab endast tarkvara arendamist ja reaalse programmkoodi kirjutamist. Füüsilist prototüübi mudeli valmistamist ja selle teoreetilist ideed käsitletakse peatükis „2. Ülevaade prototüüpmudelist”. Peatükis on välja toodud töövooalgoritm, kiire ülevaade mikrokontrollerist ja kasutamisest ning prototüüpmudelis kasutatud elementidest. Peatüki lõpus tutvustatakse lugejale lõpliku töötavat riistvaralist mudelit.

Peatükis „3. Atmel Atmega328 mikrokontrolleri programmeerimine” antakse ülevaade Arduino IDE tarkvara arendusplatvormist ning lihtsustatud kokkuvõte antud liidese tööprotsessist. Tutvustatakse arendatud töötavat tarkvara programmi prototüübi jaoks. Tutvustatakse mikrokontrolleri ja elementide vahelist suhtlemise loogikat - LCD-mooduli, UV-A LED lülitus, ventilaatori lülitus, funktsioonnuppude ja IR-anduri ning Atmega328 vahel.

Prototüübi ehitamine idee tasemest mudeli realiseerimiseni võttis väga palju aega. Mudel pole ka täna soovitud tasemeni valmis, kuid on juba näha reaalset käega katsutavat tulemust. Toote kliendini jõudmine nõuab veel viimistlemist, siiski väga palju arendust tööd on juba ära tehtud, mille käigus tuli ette ka ootamatuid olukordi ja nii reaalsete riistvaraliste moodulite kui tarkvara loogika muutusi. Antud seadme ehitamine nõuab rohkesti elektroonika produktide kasutamise manuaalide läbi töötamist ja mõistmist. Oskusteavet ja loovust nõuab ka tarkvara interpreteerimine nii isevalmistatud kui ka valmis moodulitele. Prototüübi arendamine valmis seadmeks käib edasi ka peale antud bakalaureuse töö kaitsmist. Plaanis on mikrokontroller tõsta iseseisva mikroskeemi peale ning disainida 3D prinditud korpus. Tööd ei ole plaanis hakata niivõrd turustama kuivõrd kasutusele koduseks otstarbeks, sest antud seade on veel üpris toorest, millega kaasnevad teatud kasutaja poolsed riskid.

Kasutatud kirjandus

[1] Introduction to Microcontrollers. [WWW|PDF] http://ti.tuwien.ac.at/ecs/teaching/courses/mclu/theory-material/Microcontroller.pdf, lk 1, lõik1 (18.05.2015)

[2] Light-emitting diode(LED). [WWW] http://www.electronics-tutorials.ws/diode/diode_8.html (18.05.2015)

[3] LED — a 100-year history. [WWW] http://holly.orc.soton.ac.uk/fileadmin/downloads/100_years_of_optoelectronics__2_.pdf

[4] Black Light. [RAAMAT] The Right Light: Matching Technologies to Needs and Applications, 2012, ISBN: 978-1-4398-9931, lk 108

[5] Compact fluorescent lamp (CFL). [WWW] http://whatis.techtarget.com/definition/compact-fluorescent-light-bulb-CFL (18.05.2015)

[6] Pulse Modulation of Signals. [WWW] http://web.stanford.edu/class/ee179/handouts/slide16.pdf (18.05.2015)

[7] Pulse-width Modulation - Dimming LED. [WWW] https://learn.sparkfun.com/tutorials/pulse-width-modulation (18.05.2015)

[8] Reduced Instruction Set Computer (RISC). [WWW] http://cs.stanford.edu/people/eroberts/courses/soco/projects/risc/whatis/index.html (18.05.2015)

[9] Reduced Instruction Set Computer (RISC). [WWW] http://physinfo.ulb.ac.be/divers_html/powerpc_programming_info/intro_to_risc/irt5_risc3.html (18.05.2015)

[10] Reduced Instruction Set Computer (RISC). [WWW] http://search400.techtarget.com/definition/RISC (18.05.2015

[11] Complex instruction set computing (CISC). [WWW] http://whatis.techtarget.com/definition/CISC-complex-instruction-set-computer-or-computing (18.05.2015)

[12] Instructions per second. [WWW] http://www.ibmsystemsmag.com/mainframe/tipstechniques/systemsmanagement/Don-t-Be-Misled-By-MIPS/ (18.05.2105)

[13] Analog-to-digital converter (ADC). [WWW] http://www.allaboutcircuits.com/textbook/digital/chpt-13/digital-analog-conversion/ (18.05.2015)

[14] Flash memory. [WWW|PDF] http://139.138.48.19/pdf/NAND/Toshiba/NandDesignGuide.pdf.pdf lk 4, lõik 3 (18.05.2015)

[15] Random-access memory. [WWW] http://www.pcmag.com/encyclopedia/term/50159/ram (18.05.2015)

[16] Clock rate. [WWW] http://whatis.techtarget.com/definition/clock-speed (18.05.2015)

[17] Oscillator. [WWW] http://whatis.techtarget.com/definition/oscillator (18.05.2105)

[18] Introduction to Watchdog Timers. [WWW] http://www.embedded.com/electronics-blogs/beginner-s-corner/4023849/Introduction-to-Watchdog-Timers

[19] An ape's view of the oldowan. [WWW] https://web.archive.org/web/20080716070624/http://web.uccs.edu/twynn/ApeOld.htm (18.05.2015)

[20] The first mechanical clocks. [WWW] http://www.uh.edu/engines/epi1506.htm (18.05.2015)

[21] Ultraviolet light. [WWW] http://www.pestproducts.com/uv_light.htm (18.05.2015)

[22] Atmel Atmega328. [WWW] http://www.atmel.com/devices/atmega328.aspx

[23] Atmel Atmega328.[WWW|PDF] http://www.atmel.com/Images/Atmel-8271-8-bit-AVR-Microcontroller-ATmega48A-48PA-88A-88PA-168A-168PA-328-328P_datasheet_Summary.pdf (18.05.2015)

[24] Atmel Atmega328. [WWW|PDF] http://www.atmel.com/images/Atmel-8271-8-bit-AVR-Microcontroller-ATmega48A-48PA-88A-88PA-168A-168PA-328-328P_datasheet_Complete.pdf (18.05.2015)

[25] Photoinitiator. [RAAMAT] Radiation Curing (Curopean Coatings Tech Files), 2012, ISBN: 978-3866309074, lk 31

[26] Iteaduino UNO.[WWW] http://wiki.iteadstudio.com/Iteaduino_UNO (18.05.2015)

[27] Datasheet for Iteaduino UNO.[WWW|PDF] ftp://imall.iteadstudio.com/Mainboard/IM120905006_Iteaduino_UNO/DS_IM120905006_IteaduinoUNO.pdf (18.05.2015)

[28] Schematic for Iteaduino UNO.[WWW|PDF] ftp://imall.iteadstudio.com/Mainboard/IM120905006_Iteaduino_UNO/SCH_IM120905006_IteaduinoUNO.pdf (18.05.2015)

[29] UV-A 370-380nm Datasheet. [WWW|PDF] https://avonec.de/images/3W_datasheets/370nm-380nm/3w_pcb_370nm-380nm.pdf (27.05.2015)

[30] Arduino TFT LCD Screen. [WWW] http://www.arduino.cc/en/Main/GTFT (18.05.2015)

[31] A000096 TFT LCD Datasheet. [WWW|PDF] http://www.arduino.cc/documents/datasheets/A000096_Datasheet_HTF0177SN-01-SPEC.pdf (18.05.2015)

[32] Sunon MagLev Fan. [WWW|PDF] http://media.digikey.com/pdf/Data%20Sheets/Sunon%20PDFs/ME40101V1-000U-G99.pdf (18.05.2015)

[33] TFT LCD library.[WWW|PDF] http://www.arduino.cc/en/Reference/TFTLibrary (18.05.2015)

[34] SPI library. [WWW] http://www.arduino.cc/en/Reference/SPI (18.05.2015)

[35] Serial Peripheral Interface.[WWW] https://learn.sparkfun.com/tutorials/serial-peripheral-interface-spi (18.05.2015)

[36] Mis vahe on erinevatel UV maniküüri toodetel? [WWW] http://mereneid.ee/et/nouanded_kuunetehnik/nouanded/artiklid/mis-vahe-on-erinevatel-uv-manikuuri-toodetel (18.05.2015)

[37] Seven Secrets to Properly Curing UV Gels Nails. [WWW|PDF] http://www.schoonscientific.com/downloads/tech-articles/article-7-Secrets-to-Curing.pdf (18.05.2015)

[38] The Science of Gels. [WWW] http://www.nailsmag.com/article/91808/the-science-of-gels?Page=4 (18.05.2015)

[39] UV Light Cured Gel. [WWW] http://www.nailsmag.com/article/40541/uv-light-cured-gel-how-it-works (18.05.2015)

[40] Proximity Sensor. [WWW] http://thecodeartist.blogspot.com/2011/01/proximity-sensor-on-android-gingerbread.html (18.05.2015)

[41] Spectrum sensors & controls. [WWW|PDF] http://www.digikey.com/Web%20Export/Supplier%20Content/api-technologies-1171/pdf/api-ntc-engineering-notes.pdf?redirected=1 (18.05.2015)

[42] NTC thermistor. [WWW] http://www.resistorguide.com/ntc-thermistor/ (18.05.2015)

[43] Introduction to Analog Digital Converter. [WWW|PDF] http://www.cpdee.ufmg.br/~frank/lectures/Sill-Analog-Digital-Converter.pdf (18.05.2015)

[44] UV curing. [WWW] http://www.fusionuv.com/uvlearningcenter.aspx?id=206 (18.05.2015)

[45] UV LED curing. [WWW] http://uvledcommunity.org/basics/ (18.05.2015)

[46] Arduino Uno overview. [WWW] http://www.arduino.cc/en/Main/ArduinoBoardUno (18.05.2015)

[47] ThinkPad and IdeaPad 90W Slim AC/DC Combo Adapter. [WWW] http://support.lenovo.com/us/en/documents/migr-68305 (18.05.2015)

[48] FQP50N06L N-Channel QFET ® MOSFET Datasheet. [WWW|PDF] https://www.fairchildsemi.com/datasheets/FQ/FQP50N06L.pdf. (18.05.2015)

[49] 2N3409 Datasheet. [WWW|PDF] https://www.sparkfun.com/datasheets/Components/2N3904.pdf (18.05.2015)

[50] High Power LED Driver Circuits. [WWW]

http://www.instructables.com/id/Circuits-for-using-High-Power-LED-s/?ALLSTEPS, skeem #5 (18.05.2015)

[51] LM317T Datasheet. [WWW|PDF] http://www.st.com/web/en/resource/technical/document/datasheet/CD00000455.pdf (18.05.2015)

[52] Variable Voltage Power Supply. [WWW] http://www.electronics-tutorials.ws/blog/variable-voltage-power-supply.html (18.05.2015)

[53] Controlling fan or motor speed with PWM. [WWW]

http://arduino-for-beginners.blogspot.com/2011/04/controlling-12v-fan-speed-with-pwm.htmlhttp://arduino-for-beginners.blogspot.com/2011/04/controlling-12v-fan-speed-with-pwm.html (18.05.2015)

[54] TIP122 Datasheet. [WWW|PDF] http://www.adafruit.com/datasheets/TIP120.pdf (18.05.2015)

[55] Button. [WWW] http://www.arduino.cc/en/Tutorial/Button (18.05.2015)

[56] Simple IR proximity sensor with Arduino. [WWW]

http://www.instructables.com/id/Simple-IR-proximity-sensor-with-Arduino/?ALLSTEPS (18.05.2015)

[57] TSUS5202 Datasheet. [WWW|PDF] http://www.vishay.com/docs/81055/tsus520.pdf. (18.05.2015)

[58] TSOP4138 Datasheet. [WWW|PDF] http://www.vishay.com/docs/82460/tsop45.pdf (18.05.2015)

[59] ND03 NTC Datasheet. [WWW|PDF] https://www.avx.com/docs/catalogs/nd-ne-nv.pdf (18.05.2015)

[60] Using NTC Thermistors With Arduino. [WWW] http://garagelab.com/profiles/blogs/tutorial-using-ntc-thermistors-with-arduino (18.05.2015)

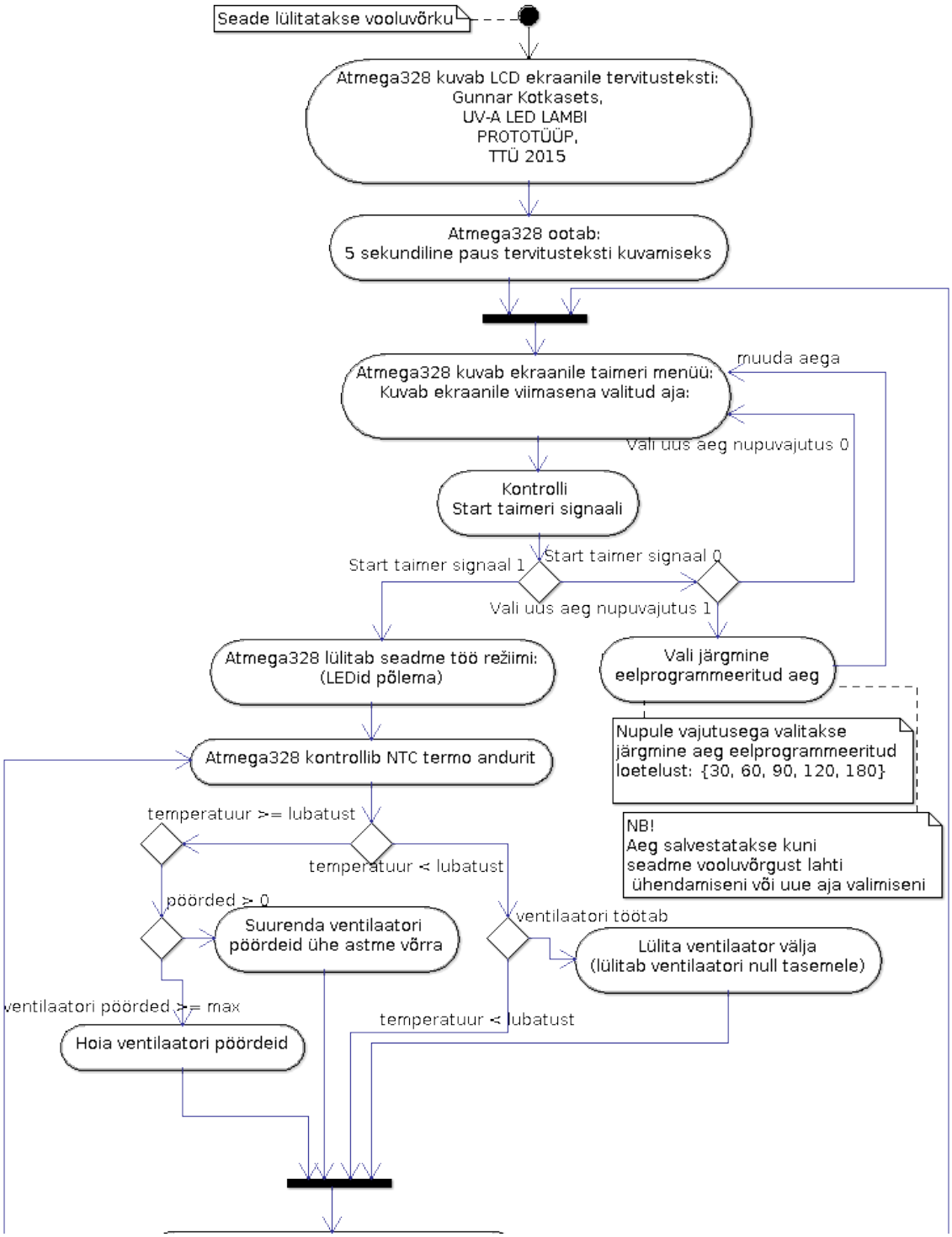
[61] Adafruit GFX Library. [WWW] https://github.com/adafruit/Adafruit-GFX-Library (18..05.2015)

[62] Adafruit ST7735 Library. [WWW] https://github.com/adafruit/Adafruit-ST7735-Library. (18.05.2015)

[63] How to use infrared receiver sensors for collision avoidance. [WWW]

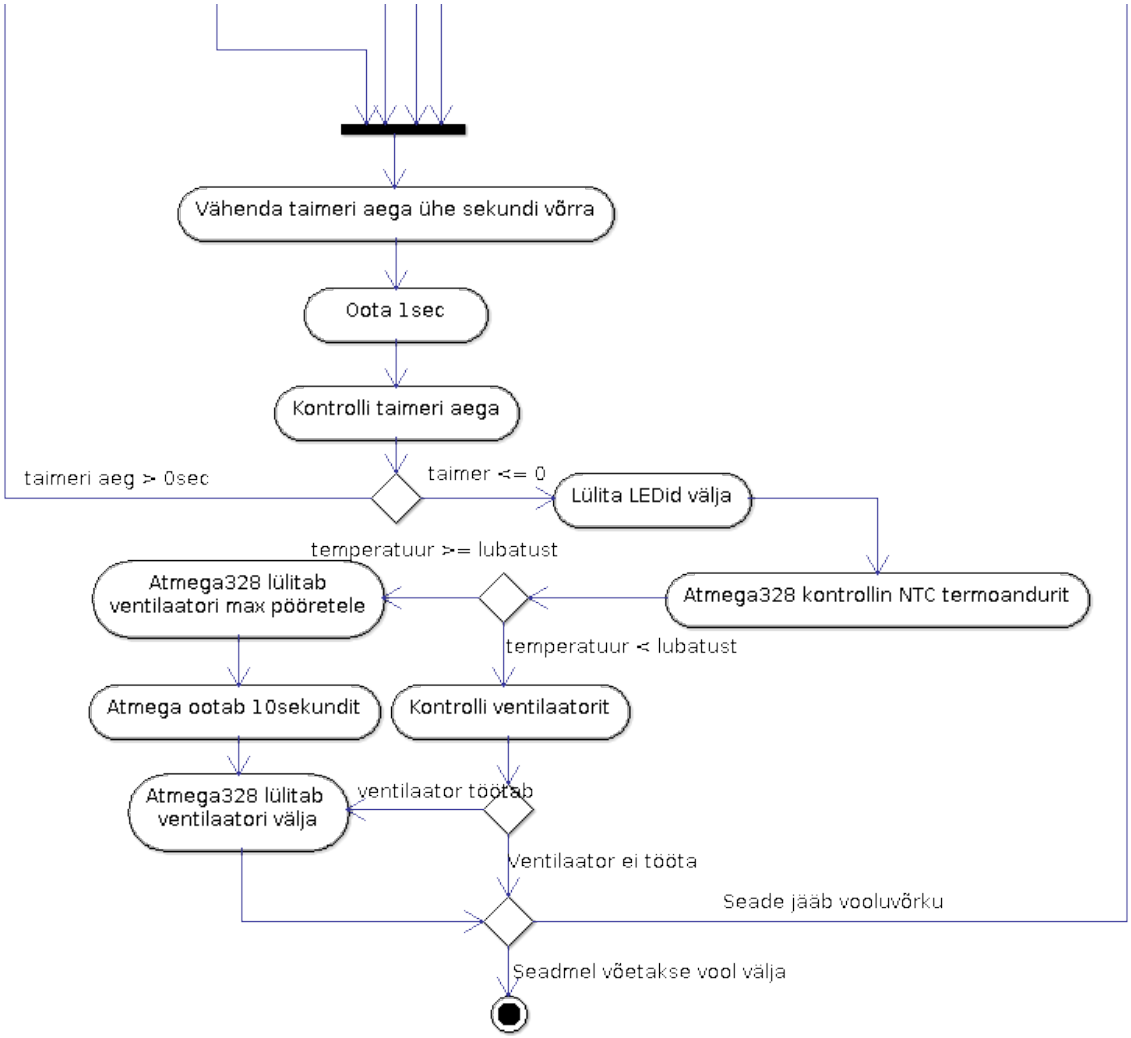
http://letsmakerobots.com/content/how-use-infrared-receiver-sensors-collision-avoidance (18.05.2015)

Lisa 1. Prototüüpmudeli töövoo täielik tegevusalgoritm



Lisa 1 jätkub järgmisel lehel...

...**Lisa 1** jätk.



Lisa 2. Prototüüpmudeli täielik tarkvara lähtekood

```
/*
 * @firmware: UV-A LED Lamp On Atmega328 Microcontroller For Gel Lack Polymerisation
 * @version: 0.1
 * @author: Gunnar Kotkasets
 * @date: 23.04.2015
 * @description: This version of firmware is very first release and not passed extended testing, so be aware of using this firmware.
 */

////[teegid]

#include <TFT.h> //Arduino LCD library
#include <SPI.h> //SPI

////[mikrokontrolleri defineeritud pordid]

/* What pin are used | Kasutusel olevad mikrokontrolleri pordid */
// pin definition for Iteaduino Uno
// define pin 13 [LCD   SCK]
// define pin 12 [LCD  MISO]
// define pin 11 [LCD  MOSI]
#define cs 10 // [LCD    CS]
#define dc 9  // [LCD   D/C]
#define rst 8 // [LCD RESET]
#define selectButton 7     //[menu select button / menüü "Vali aeg"]
#define IRledPin 6         //[]
#define fanPWM 5           //[used - fan PWM]
#define IRsensorPin 4      //[used - IF using sd card]
#define ledPWM 3           //used - LED PWM]
#define stopTimerSignal 2  //[used - start button]
```

```
//define pin 1          //[free / vaba]

//define pin 0          //[free / vaba]


/* END OF PIN definition / mikrokontrolleri portide
defineerimise lõpp*/


////[termotakisti konstandid]


/* NTC thermistor constants / NTC termotakisti
konstandid*/

#define NTC_PIN A0              //Analog input pin, read
thermistor / Mikrokontrolleri analoog port A0

#define NTC_RESISTANCE 10000    //NTC thermistor
resistance in ohms / Termotakisti pull-down takisti
väärtus 10000ohm

#define NTC_NOMINAL_TEMP 25     //Temperature that
NTC thermistor is 10000ohm / Termotakisti nominaal
takistus on 25C juures

#define NTC_NR_OF_SAMPLES 5     //Number of
samples / Mitu mõõtmist teha ühes tsüklis

#define NTC_BETA_COEFFICIENT 4080 //NTC thermistor
beta value (3977) / Termotakisti Beta koefitsient 4080

#define NTC_RESISTOR 10000      //Series resistor with
NTC thermistor / Termotakisti enda väärtus 25C
temperatuuri juures 10000ohm


////[termotakisti muutujad]


/* NTC thermistor variables / Programmi töö kestel
muutuvad termotakisti parameetrid */

int NTCSampleArray[NTC_NR_OF_SAMPLES]; //NTC
samples array / NTC termotakisti mõõtmiste tulemuste
massiiv ühes tsüklis

float Temperature;              //Current temperature /
Hetke mõõdetud temperatuur

String LastTempValue = "0.00";      //Last temperature
value / Viimat mõõdetud ehk eelmine temperatuur

char TempSenorPrintOut[6];          //Print temperature
value as char array to LCD / LCD jaoks loodud massiiv
temperatuuri kuvamiseks ekraanile.


////[programmi muutujad]
```

```
//Status variables / Programmi käigus muutuvad staatused

int selectButtonStatus = 0;     //Select Button status / Menüü "Vali aeg" staatuse muutuja

int startTimerSignalStatus = 0; //Start timer signal status / Taimeri loenduse start signaali staatuse muutuja

bool fanStatus = false;         //Fans running status / Ventilaatori sees ja väljasoleku staatus

int fanRPM = 0;                 //RPM of fan / Ventilaatori pöörete arv skaalal 0-255


/*

 * define colors in RGB:

 * PINK: 255, 0, 204

 * BLUE: 0, 0, 255

 * PURPLE?: 255, 28, 128

 * BLACK: 0, 0, 0

 */

const int textColor[] = {255, 0, 204}; //Default text color / teksti värv 1

const int textColor2[] = {0, 255, 0 }; //Default text color2 / teksti värv 2

const int bgColor[] = {0, 0, 0};       //Default background color /tagatausta värv


/*

 * define text size / Teksti suurus

 */

const int timerTxtSize = 7;//Timer text size / Taimeri teksti suurus

const int txtSize = 2; //Default text size / Tavalise teksti suurus


////[TFT LCD instants]


//create an instance of the library / TFT LCD ekraani instantsi loomine

TFT TFTscreen = TFT(cs, dc, rst);
```

char timerCharArray[4]; //timer char array for printing current time left on screen / ajaloenduri char massiiv järele jäänud aja LCD ekraanile printimiseks

const int timeSelectOption[] = { 30, 60, 90, 120, 180 }; //Time selection for countdown / Taimeri menüü eel-defineeritud ajad

int index = 0; //default index of timer menu / Menüü ajavaliku massiivi indeks, mis kuvatakse ekraanile mikrokontrolleri käivitamisel

int timerCurrentTime = timeSelectOption[index]; //Current start time / Mikrokontrolleri käivitudes kuvatakse aeg menüüst

////[programmi eelseadistus setup()]

//set-up on boot / Mikrokontrolleri käivituse eelseadistus

void setup() {

  Serial.begin(9600);

  analogReference(EXTERNAL);

  // put your setup code here, to run once:

  pinMode(selectButton, INPUT); //Select menu as input / "Vali aeg" määra sisend pordiks

  pinMode(stopTimerSignal, INPUT); //Stop timer signal as input / Taimeri stop signaal nupp vali sisend pordiks

  pinMode(ledPWM, OUTPUT); //Led PWM signal as OUTPUT/ Ledide pwm signaal

  pinMode(fanPWM, OUTPUT); //Fan PWM signal as OUTPUT/ Ventilaatori PWM signaal

  pinMode(IRledPin, OUTPUT); //IR LED as OUTPUT / IR ledid väljund signaal

  setupDisplay(); //Setup display method / Algseadista LCD ekraan

}

////[LCD eelseadistus]

void setupDisplay()//Setup display method / Algseadista LCD ekraan

{

```
  TFTscreen.begin();   //Turn on screen / initsialiseeri ekraan


  TFTscreen.background(bgColor[0], bgColor[1], bgColor[2]); //Set background black / Seadista tagatausta värv


  //welcome();

  ttyWelcome();   //Print welcome screen / Tervitustekst


  delay(5000); //Delay of welcome tekst / Tervitusteksti kuvatakse 5sek


  TFTscreen.background(bgColor[0], bgColor[1], bgColor[2]);   //Set background black / Kustuta kõik ekraanilt

}
```

////[Programmi elutsükkel]

```
//loop on run / tsükkel mida täidetakse pidevalt mikrokontrolleri tööl oleku ajal

void loop() {

  // put your main code here, to run repeatedly: / Siia pannakse mikrokontrolleri programmi elutsükkel

  selectTime(); // Select time menu method / Aja valimise menüü meetod


  timer(timerCurrentTime); //Timer countdown and working method / Taimeri ajalugemise ja töörežiimi meetod

}
```

////[Tervitusteksti meetod]

```
//welcome screen / Teretulemast default 1 tervitus tekst

void welcome()

{

  TFTscreen.stroke(textColor[0], textColor[1], textColor[2]);   //Text color / Teksti värv
```

```
  TFTscreen.setTextSize(txtSize);   //Text size / Teksti
suurus


  TFTscreen.text("W e l c o m e", 3, 55); //Welcome text /
Tervitus tekst


  TFTscreen.setTextSize(txtSize); //Set text size /
Seadista teksti suurus tagasi

}


//welcome screen / Tervitustekst kooli jaoks

void ttyWelcome()

{

  TFTscreen.stroke(textColor[0], textColor[1],
textColor[2]); //Text color / Teksti värv


  TFTscreen.setTextSize(txtSize); //Text size / Teksti
suurus


  /* Welcome tekst message / Tervitus tekst*/

  TFTscreen.text("Gunnar", 45, 0);

  TFTscreen.text("Kotkasets", 25, 20);

  TFTscreen.text("UV-A", 0, 40);

  TFTscreen.text("LED LAMBI", 0, 60);

  TFTscreen.text("PROTOTYYP", 0, 80);

  TFTscreen.text("2015", 0, 100);


  TFTscreen.setTextSize(txtSize); //Set text sixe /
Seadista teksti suurus tagasi vaikeväärtuseks

}
```

////[Lambi menüü meetod]

```
/* Select time method / "Vali aeg" menüü meetod */

void selectTime()

{

  int lastScreen = 0; // Last screen time / viimane tekst
menüüs ekraanil
```

```
  TFTscreen.setTextSize(txtSize);   //Set text size to
normal / Seadista teksti suurus


  strokeScreen(false);   //Set text color / Seadista värv
teksti kustutamiseks


  TFTscreen.text("Select time:", 0, 0); // Select time /
kuvatakse tekst "Vali aeg" ekraanil


  do { //do while cycle / tsükkel tee enne ja siis kontrolli


    selectButtonStatus = digitalRead(selectButton); //
Read select button status / Loe menüü "Vali aeg" nupu
staatust


    SendIRSignal();//Send IR signal / Saada IR LEDidele
signaal, et kontrollida käe sisestamiset

    if (digitalRead(IRsensorPin) == LOW) // Read id IR
sensor is LOW / loe IR sensorilt madal signaali

    {

      startTimerSignalStatus = 1; //Set start timer signal
status 1 / Seadista start timeri signaal üheks

      Serial.println("IRSensorPin: HIGH"); //Debug signal /
logi serial monitori väärtus

    }

    else // Id IR sensor signal high / kui IR sensori signaal
on kõrge

    {

      startTimerSignalStatus = 0; //Set start timer signal
LOW / Seadista start taimeri signaal nulliks

      Serial.println("IRSensorPin: LOW"); //Debug signal /
logi serial monitori väärtus

    }

    delay(10); //delay 10ms / oota 10ms


    if (selectButtonStatus == HIGH) //Select menu button
HIGH / Kontrolli kas valitakse uus aeg nupuga "Vali aeg"
menüüst

    {

      delay(500); //Wait delay 500ms / Oota programmi
sujuvamaks jooksmiseks 500ms
```

```
      if ((index + 1) >= (sizeof(timeSelectOption) /
sizeof(*timeSelectOption))) //Check if array is in the end
and select index of start array / vali aja menüü alg aeg,
kui menüü sai läbi

        index = 0;

      else //Select next time / Vali indeksiga uus aeg
menüüst

        index++;

    }


    timerCurrentTime = timeSelectOption[index]; //Save
time from array as variable / Salvest massiivist järgmine
aeg kui uus muutuja


    if (lastScreen != timerCurrentTime) //Print screen
logic / Kuva uus aeg ekraanile LCD kuvamise loogika

    {

      printTimer(lastScreen, true); //Print timer / prindi
taimeri aeg

      delay(100); //Delay 100ms / oota 100ms

      printTimer(timerCurrentTime, false); //Delet time
from screen / Kustuta eelmine aeg ektraanilt

      lastScreen = timerCurrentTime; //Save current time
as last time for next cycle / Salvesta hetke aeg kui
eelmine aeg järgmise tsükli jaoks

    }

  }

  while (startTimerSignalStatus == LOW); // Cycle as
Start timer signal is LOW / Korda tsüklit niikaua kui pole
antud taimeri start käsku


  selectButtonStatus = 0; // Set select Button status as 0
again / Seadista menüü nupu staatus nulliks peale
menüü tsükli lõppu

  startTimerSignalStatus = 0; // Set Start timer signal
status status as 0 again / Seadista Start taimeri signaali
staatus nulliks peale menüü tsükli lõppu


  delay(500); //Delay 500ms / oota 500ms


  TFTscreen.setTextSize(txtSize);   //Text size / Seadista
teksti suurus
```

```
  strokeScreen(true);   //Set text color / Seadista teksti
värv


  TFTscreen.text("Select time:", 0, 0);   //Clear text /
kustuta tekst


  delay(100); //Delay 100ms / oota 100ms

  strokeScreen(false);  //Set text color / seadista värv
teksti kustutamiseks

}
```

////[Lambi töörežiim ehk taimeri meetod]

```
//Timer method / Timeri ja töörežiimi meetod

void timer(int secondsToCountDown)

{


  turnLED(255); //When method starts turn UV-A LED ON
/ Kui meetodisse sisenetakse siis lülita UVA ledid kohe
sisse

  for (int counter = secondsToCountDown; counter >=
0; counter--) //Start counter countdown / ALusta taimer
maha lugemist

  {

    getTemperature(); //Check temerature / KOntrolli
temperatuuri

    if (Temperature > 27 && fanRPM < 255) //Check if
FAn needs to turn on / Kontrolli kas ventilaator peab
sisse lülitatama

      turnFAN(true); //Turn fan on

    else if (Temperature <= 27 && fanRPM > 0)//Check if
FAn needs to turn off / Kontrolli kas ventilaator peab
välja lülitatama

      turnFAN(false); //Turn fan off / Lülita ventilaator välja

    printTimer(counter, false);//print timer text / Prindi
taimeri tekst

    delay(1000); //Delay 1000ms -> 1s / Oota tsükliga 1
sekund

    printTimer(counter, true); //delete last text / Prindi
taimeri eelmine tekst

    if (digitalRead(stopTimerSignal) == HIGH) //Read
stop interupt value / Loe taimeri stop signaali kasutaja
poolt
```

```
    {
      TFTscreen.setTextSize(txtSize); //Text size / Set text
size


      strokeScreen(false); //text color / Muuda teksti värvi


      TFTscreen.text("Stop...", 30, 55); // Print Stop text /
Prindi tekst ekraanile, et taimer petatakse


      delay(1000); //Wait 1s / oota üks sekund


      TFTscreen.background(bgColor[0], bgColor[1],
bgColor[2]); //Set background black / puhasta ekraan
      delay(1);//Delay 1ms just in case / Otta 1ms
igaksjuhuks


      turnLED(0); //Turn led off / Lülita UV-A ledid välja


      coolingFan(); //Check if Fan needs to bee cooled /
Kontrolli kas on vaja ventilaator jahutamist peale tsükli
lõppu


      turnFAN(false); // lülita ventilaator välja kui see pole
veel väljalülitatud


      return; //Return cycle / välju tsüklist
    }
    delay(100); //WQait 100ms just in case / oota 100ms
igaks juhuks parema Stop taimer signaal püüdmiseks
  }
  turnLED(0); //Turn LED off / Lülita ledid välja normaalse
tsükli lõpust ehk taimeri lõpetamisel


  coolingFan(); //Check if Fan needs to bee cooled /
Kontrolli kas on vaja ventilaator jahutamist peale tsükli
lõppu


  turnFAN(false);// lülita ventilaator välja kui see pole
veel väljalülitatud


  TFTscreen.background(bgColor[0], bgColor[1],
bgColor[2]);   //Set background black / Puhasta ekraan
```

```
  TFTscreen.setTextSize(txtSize); //   //Text size / seadista
uus teksti suurus


  strokeScreen(false); //text color / seadista teksti värv


  TFTscreen.text("Finished!", 30, 55); // Print text / Prindi
tekst

  strokeScreen(true); //clear text color / Teksti
kustutamise värv

  delay(5000); //Delay 5s / oota 5s

  TFTscreen.text("Finished!", 30, 55); //Clear tekst after
5s / Kustuta tekst peale 5s ootamist

}
```

////[Ekraanile taimeri kuvamise meetod]

```
//print timer out / method that prints timer current
status / Taimeri hetke aja printimise meetod

void printTimer(int second, bool clrScreen)

{

  String secondVal = String(second); //timer seconds to
string / taimeri sekundid stringiks

  secondVal.toCharArray(timerCharArray, 4); //timer
seconds to array string / taimeri sekundid char
massiiviks


  strokeScreen(clrScreen); //Print or clear screen or not /
vastavalt parameetrile vali värv ekraani puhastamiseks
või värvimiseks


  TFTscreen.setTextSize(timerTxtSize); //Text size /
seadista teksti suurus


  /* Change timer seconds to center of display /

   * muuda taimeri sekundite asukohta ekraanilt
vastavalt 1, 2, 3 kohaline numbrile

   */

  if (second >= 100)

    TFTscreen.text(timerCharArray, 20, 40);
```

```
  if (second >= 10 && second < 100)
    TFTscreen.text(timerCharArray, 42, 40);

  if (second < 10)
    TFTscreen.text(timerCharArray, 64, 40);
}
```

////[Ekraani teksti värvimise abi meetod]

```
//change text color and clear / meetod kirja värvi valimiseks kustutamise või printimise jaoks
void strokeScreen(bool clrScreen)
{

  if (clrScreen) //Text color / kui kustuta siis vali tagatausta värv
    TFTscreen.stroke(bgColor[0], bgColor[1], bgColor[2]); //clear text / kustuta tekst
  else
    TFTscreen.stroke(textColor[0], textColor[1], textColor[2]); //print text / Kui prindi siis vali tavaline värv
}
```

////[UV-A LED juhtimine]

```
/*Turn LED on method / Meetod millega UV-A ledid sisse lülitatakse*/
void turnLED(int brightness)
{
  analogWrite(ledPWM, 0); //pre set to 0 / algseadista ledide heledus 0 tasemele

  if (brightness == 0) //If brightness is 0 / kui heledus == 0
  {
    analogWrite(ledPWM, brightness); //write led PWM 0 / Lülita LEDid välja
  }
  else //If brightness is not 0 / kui heledus != 0
  {
```

```
   for (int counter = 0; counter <= brightness;
counter++) //Smooth LED turn on / Lülita ledid sujuvalt
sisse

   {

     analogWrite(ledPWM, counter); //Turn led step by
step / lüluta led PWMiga järkjärgult heledamaks

     delay(5); //delay 5ms / oota iga tspkli järgi 5ms

   }

  }

}
```

////[Ventilaatori juhtimine]

```
/*Turn fan on or off / lülita ventilaator sisse või välja*/

void turnFAN(bool turnOn)

{

  if (turnOn == false) //Turn fan off / lülita ventilaator
välja

  {

   fanStatus = false; //Set fanStatus false / seadista
ventilaatori staatus väljalülitatuks

   fanRPM = 0; //Set rpm to 0 / seadista ventilaatori
pöörded 0ks

   analogWrite(fanPWM, fanRPM); //Turn FAN off with
PWM / Lülita ventilaator PWMiga välja reaalselt

  }

  else

  {

   fanStatus = true; //Set fan status true / seadista
ventilaatori olek töötab


   if (fanRPM <= 235) //IF fan rpm is under value on
PWM scale 100 / kui ventilaatori pöördeid on PWM
skaalal alla 100

     fanRPM = 235; //set fan rpm as / seadista
ventilaatori pööreteks 100 PWM'i

   else if (fanRPM >= 235 && fanRPM <= 255) //* * *

     fanRPM += 10; //* * *

   else if (fanRPM >= 255) //* * *

     return;
```

```
    analogWrite(fanPWM, 255); //Write fan speed as PWM scale / Kirjuta ventilaatorile kiirus PWM skaalas.
  }


  delay(10); //delay 10ms / oota 10ms igaksjuhuks
}
```

////[Ventilaatori jahutamine]

```
/* Method for cooling fan */
void coolingFan()
{
  if (fanStatus == true) //If fan is workin / Kui ventilaator töötab
  {
    TFTscreen.setTextSize(txtSize);     //Text size / Teksti suurus

    strokeScreen(false); //print text // teksti printimiseks värv
    TFTscreen.text("Cooling...", 30, 55); //Print as cooling / Prindi jahutan
    strokeScreen(true); //clear text / Valiv värv teksti kustutamiseks

    delay(10000);//Wait 10s for cooling / oota 10 sek jahutamiseks
    TFTscreen.text("Cooling...", 30, 55); //clear text / kustuta tekst jahuta
  }
}
```

////[Temperatuuri lugemine]

```
/*Method for get current temperature*/
void getTemperature()
{
  float TempMedian; //Float variable for temp average / Temperatuuri mediaan keskmine
  int i; //for cycle counter / Tsükli loendaja
```

```
  /* takse samples / võta temperatuuri proove */
  for (i = 0; i < NTC_NR_OF_SAMPLES; i++) {
   NTCSampleArray[i] = analogRead(NTC_PIN);
   delay(10);
  }

  /* Add samples / liida proovid omavahel */
  TempMedian = 0;
  for (i = 0; i < NTC_NR_OF_SAMPLES; i++) {
   TempMedian += NTCSampleArray[i];
  }
  /* Calculate median / Arvuta keskmine temperatuur */
  TempMedian /= NTC_NR_OF_SAMPLES;
  // Convert the thermal stress value to resistance
  TempMedian = 1023 / TempMedian - 1;
  TempMedian = NTC_RESISTOR / TempMedian;

  /*Calculate temperature using the Beta Factor
equation / Arvuta ja konverteeri temperatuur
Celsiusesse*/
  Temperature = TempMedian / NTC_RESISTANCE;     //
(R/Ro)
  Temperature = log(Temperature); // ln(R/Ro)
  Temperature /= NTC_BETA_COEFFICIENT;                  //
1/B * ln(R/Ro)
  Temperature += 1.0 / (NTC_NOMINAL_TEMP + 273.15);
// + (1/To)
  Temperature = 1.0 / Temperature;                // Invert
the value
  Temperature -= 273.15;                       // Convert it to
Celsius

  //Debug / silu
  Serial.print("The sensor temperature is: ");
  Serial.print(Temperature);
  Serial.println(" *C");

  String SensorValue = LastTempValue; //Last temp
value / eelmine temperatuuri näit
```

```
  SensorValue.toCharArray(TempSenorPrintOut, 6); //To
char array / printimiseks massiivi temperatuuri väärtus


  TFTscreen.setTextSize(txtSize);  //Text size / seadista
teksti suurus


  TFTscreen.stroke(bgColor[0], bgColor[1],
bgColor[2]); //Set color / seadista tekstivärv

  TFTscreen.text(TempSenorPrintOut, 0, 0);//kustuta text
temperature / kustuta eelmine temperatuuri tekst
ekraanilt


  LastTempValue = SensorValue =
String(Temperature); // Save new last temp / salvesta
uus eelmine temperatuur


  SensorValue.toCharArray(TempSenorPrintOut, 6);  //To
char array / printimiseks massiivi temperatuuri väärtus


  TFTscreen.stroke(textColor2[0], textColor2[1],
textColor2[2]); // Set color / seadista värv uue
temperatuuri printimiseks


  TFTscreen.text(TempSenorPrintOut, 0, 0); //Print new
temp / prindi uus temperatuur

}
```

////[IR anduri jaoks IR signaali saatmine]

```
/*Method for sending IR signal to fototransistor / meetod
ir signaali saatmiseks*/

void SendIRSignal()

{

  for (int i = 0; i <= 384; i++) { //count cycle 384
times / loenda 384 korda

    digitalWrite(IRledPin, HIGH); //print 13ms as high ir
led / lülita 13ms sisse ir led

    delayMicroseconds(13);

    digitalWrite(IRledPin, LOW); //print 13ms as high ir led
/ lülita 13s välja ir led

    delayMicroseconds(13);  } }
```

Lisa 3. UV-A lambi prototüübi pilt