1918
TALLINNA TEHNIKAÜLIKOOL

INFORMAATIKATEADUSKOND
Thomas Johann Seebecki elektroonikainstituut

Köögikubu juhtimine mikrokontrolleri baasil

Microcontroller-based kitchen hood control

Üliõpilane: Alfred Hiie

Juhendaja: dots. Mihhail Pikkov

Tallinn, 2015.a.

SISUKORD

TALLINNA TEHNIKAÜLIKOOL

Thomas Johann Seebecki elektroonikainstituut

BAKALAUREUSETÖÖ ÜLESANNE

Alfred Hiie, üliõpilaskood 130010 IAEB

Bakalaureusetöö teema:
(eesti keeles) Köögikubu juhtimine mikrokontrolleri baasil
(inglise keeles) Microcontroller based kitchen hood control

Ülesanne: Tööstuslikele köökidele mõeldud köögikubu juhtsüsteemi automatiseerimine.

Lahendamisele kuuluvate probleemide loetelu:

1. Süsteemi ja algoritmi koostamine
2. Andurite ja komponentide valik
3. Andurite testimine ja kalibreerimine
4. Juhtsüsteemi koostamine

Lõputöö eesmärgid:
Koostada temperatuuri-, gaasi- ja valgusandurite abil toimiv juhtsüsteem köögikubule. Testida andurite tööd ning nende näidud panna vastavusse juhtalgoritmiga. Programmeerida mikrokontroller tagasiside andmiseks ja juhtalgoritmi täitmiseks. Optimeerida suurköökide energia kulu.

Bakalaureusetöö esitada instituuti eesti keeles 2 eksemplaris hiljemalt 1. juunil 2015.a.

Juhendaja:	Ülesande vastu võtnud:
Dotsent M. Pikkov	Üliõpilane A. Hiie .....

AUTORIDEKLARATSIOON

Deklareerin, et käesolev lõputöö on minu iseseisva töö tulemus.

Esitatud materjalide põhjal ei ole varem akadeemilist kraadi taotletud.

Töös kasutatud kõik teiste autorite materjalid on varustatud vastavate viidetega.

Kuupäev: 08.06.2015

Töö autor: …….

/allkiri/

EESSÕNA

Töö arenes välja ventilatsiooniseadmeid ja –tarvikuid tootva firma ETS Nord vajadusest luua energiasäästlik lahendus restoranide ja muude toitlustusasutuste köökide ventileerimiseks. Kuna autor ise peale eduka praktikumi läbimist antud firma arendusosakonnas tööle asus ja antud teema vastu huvi tundis, siis sellest tuli ka antud teema valik. Kuna tegemist on ettevõttele vajaliku tööga, siis leiab see peale valmimist reaalset kasutust. See teeb sellest palju atraktiivsemaks ülesande võrreldes teistega, millel puudub kindel tuleviku ootus.

Konsultatsioonide andmise ja juhendamise eest tahab autor tänada eelkõige Tallinna Tehnikaülikooli dotsenti Mihhail Pikkovit, kuid ka ettevõtte ETS Nord arendus- ja juhtimisosakondasid.

Lühendid ja mõisted

1. Arduino raamatukogu – Arduino kommuuni poolt loodud koodid, mis võimaldavad lihtsustatult kasutada teatud mikrkokontrolleri funktsioone ; i.k. Library
2. LED Valgust kiirgav diood; i.k Light-Emitting Diode
3. MCU Mikrokontroller; i.k. Microcontroller Unit
4. Flash - Välkmälu
5. GND – elektriline maapotensiaal ehk 0V maandus; i.k. Ground
6. SDA – I²C protokollil toimiv andmesiin ; i.k. Serial Data Line
7. SCL – I²C protokollil toimiv kellataktisiin ; i.k. Serial Clock Line
8. PWM – Pulsi pikkuse modulatsioon. Määrab ära aplikatsiooni töötsükli pikkuse ; i.k. Pulse Width Modulation
9. MOSFET – Isoleeritud paisuga väljatransistor, milles tüüritava elektriväljaga muudetakse laengukandjate kontsentratsiooni kanalis ; i.k. Metal–Oxide–Semiconductor Field-Effect Transistor
10. Säästurežiim – Arduino mikrokontrolleritel on sisse ehitatud võimalus panna kontroller nö magama ehk olla valmis tööks, kuid samal ajal tarvitada võimalikult vähe voolu; i.k. Sleep Mode
11. Kõrge (madal) signaal – loogiline signaali väärtus 1 (0) ehk pingenivoo ulatub üle kriitilise piiri; i.k. Logical High (Low)

SISSEJUHATUS

21. sajandil oleme jõudnud olukorda, kus arenenud riikides on elekter saadaval peaaegu kõikjal ning seetõttu kasutatakse seda optimaalselt suuremates kogustes. Kuna siiani toodetakse elektrit enamasti veel otsa lõppevatest maavaradest, siis on energia säästlikkus kindlasti üks olulisemaid kõneaineid. Niikaua kuni alternatiivsetest energiaallikatest saadav elektri kogus ei ole piisav kogu maailma energia kulude korvamiseks, tuleks üritada vähendada energia tarbimise vajadust nii palju kui võimalik. Siit ongi antud lõputöö projekti eesmärk eelkõige vähendada tööstuslike köökide energiakulu ning luua köögikubu, mis toimiks võimalikult ökonoomselt ja loodussõbralikult. Antud lahendus lihtsustab ka kokkade ja teiste köögis töötavate isikute tööd seeläbi, et nad ei pea muretsema selle eest, et töökeskkond ei oleks optimaalselt ventileeritud. Lisaks on süsteemi sisse integreeritud ka suitsuanduri funktsioon, tänu millele peaksid töötajad olema võimelised likvideerima tulekahju enne, kui selle levik peatamatuks muutub.

Töö käiku on kõige parem vaadelda lähtuvalt kuuest suuremast etapist:

Töökeskkonna nõuetega tutvumine ja parameetrite määramine

Süsteemi välja mõtlemine. Skeemi ja mudeli loomine.

Andurite testimine ja kalibreerimine.

Juhtalgoritmi koostamine

Prototüübi loomine ning selle testimine.

Valmislahenduse projekteerimine.

Töö esimese punktina pidi autor ennast kurssi viima vastavate töökeskkonna nõuetega ning sealsete töötajate vajaduste ja harjumustega. Sealt tuli välja, et enamus kokkadest kas ei pööra ruumi ventileerimisele üldse tähelepanu või lasevad süsteemil toimida maksimaalse võimsusega terve tööaja. Siit tulenevalt sai ka autor kinnitust, et automatiseeritud süsteem oleks seega väga vajalik enamustesse suurköökidesse. Ega enamus töötajad ei ole ventilatsioonispetsialistid ning ei peagi teadma, kui suur peaks olema sissetõmbe ja väljapuhke õhuhulgad. Paratamatult säilib aga mõttelaad „parem karta, kui kahetseda“ ning seega hoitakse ventilaatoreid pidevalt liigse võimsusega töös. Üleliigsel ventileerimisel ei ole mingeid hüve ei töötingimuste ega ka mitte valmivate toitude koha pealt. Sellised probleemid kaasnevad hoopis juhul, kui ventilatsioon ei tööta piisava võimsuse juures. Sellist probleemi siiski väga tihti ei kohta, vähemalt mitte autori kogemuste kohaselt. Liigne ventileerimine on seevastu palju aktuaalsem probleem, mis toob endaga kaasa mõttetu energia kulu ning kiired õhuvoolud võivad tuua kaasa muid ohte või anomaaliaid köögi toimkonna igapäeva tegevuses.

Peale sellist tagasisidet sai selgeks, et antud projekt on väga vajalik. Seega asus autor süsteemi koostama. Esmalt sai tehtud üldskeem, mis näitab mõtlikult süsteemi struktuuri ning selle tööd. Selle põhjal oli hiljem väga hea luua Eagle tarkvaral elektriskeemid nii erinevatele anduritele kui ka täissüsteemile. Andureid sai lihtsuse ja mugavuse mõttes testitud eraldi. Testprogramme sai jooksutatud kuni 24h pikkuselt. Seda mitte ainult seetõttu, et osad kiirrestoranid on ööd läbi lahti, vaid kuna osades töökohtades sai ilmsiks, et mõnda toitu on vaja nii pikalt valmistada, et pliidid jäetakse terveks ööks tööle. Selline informatsioon andis kinnitust ka suitsuanduri vajadusele.

Andurite testimisel tuli ka ette mitmeid katsumusi, kuid sellest kõigest tuleb täpsemalt juttu töö käigus. Bakalaureusetöö tegemisel kasutati Arduino tarkvara, kuna testitavad mikrokontrollerid olid vastavalt valitud ning antud keskkond on tehtud kasutajale palju mugavamaks võrreldes enamus teiste võimalustega. Antud tarkvara toimib natukene modifitseeritud C++ programmeerimiskeelel.

PÕHIOSA

1. Töökeskkonna nõuded ja parameetrid

Iga tööandja või ettevõtja kohustus on läbi viia töökeskkonna riskianalüüs. Selle kägius tehakse kindlaks töökeskkonna ohud ning vajadusel mõõdetakse nende parameetreid. Antud keskkonnas, tööstuslikus suurköögis, on neid meeletu hulk, kuid me puudutame vaid neid, mis selle töö raamesse jäävad.

Töökeskonna ohutegurid:

Värske õhk – kindlasti on antud töö puhul kõige tähtsamaks punktiks tööruumi vajalik ventileerimine. Siin kohal peavad olema piisavad nii sissepuhuva kui ka välja tõmmatava õhu hulk. Esimene seepärast, et töötajatel piisavalt puhast õhku jätkuks ning väljatõmme, kuna toidu valmistamisel eralduvad kuumad aurud on samuti töötajate tervisele kahjulikud.

Temperatuur – enamasti on köökide temperatuur liiga kõrge, kuna peale sisse ehitatud kütte, soojendavad seda ruumi ka mitmed pliidid, ahjud ja muud soojusallikad. Siit tulenevalt on süsteemi ülesandeks tavaliselt kuumad õhuvoolud välja juhtida ning uued jahedad peale tuua. Siiski ei saa sisse tuua liiga külma õhku, eriti talvel. Seepärast on kubusse sisse ehitatud ka soojusülekande funktsioon, mille läbi soojendatakse sisse puhuvat õhku välja tõmmatava õhu soojuse arvelt. Kuumematel suvepäevadel, kui väline temperatuur ületab töökeskkonna oma, muutub see protsess vastupidiseks.

Valgustus – kuna kubu katab köögis põhi tööala, siis on selle valgustamine väga oluline punkt. Paika on pandud kindlad normid, mida peab valgustuse valikul järgima. Nagu ka eelnevate punktide puhul, peab valgustus olema optimaalne. Pimedas töötamine on väsitav ja silmadele kahjulik, kuid kui valgustus on liiga intensiivne või liiga kindlalt fokuseeritud, siis võib see läikivatelt tööpindadelt tagasi hakata peegelduma.

Müra ja vibratsioon – kui ventilaatorid töötavad maksimaalsel võimsusel, siis tekitavad nad tugevat müra, mis ventilatsioonisüsteemis edasi kandub. See võib omakorda võimenduda ventilatsioonitorudes, kus õhu liikumise kiirus on liiga suur. Seal tekivad hõõrde ja konstruktsiooni nihkumise tõttu edasisi mürasid, mis teistega liituvad. See on samuti üks olulisi ajendeid ventilaatorite kiiruse optimeerimiseks.

Õhu puhastus – keskkonna nõuetest tooks välja veel õhu puhastuse tähtsuse. Seda tehakse välja mineva õhuga, mis esmalt lastakse läbi tsüklofiltrite, kus suurem osa rasva ladestub ning ära voolab, ning seejärel suunatakse see läbi osoonigeneraatori. Osoon peaks suutma puhastada õhust peaaegu kõik rasvaosakesed ja muud kahjulikud gaasid.

2. Mikrokontroller

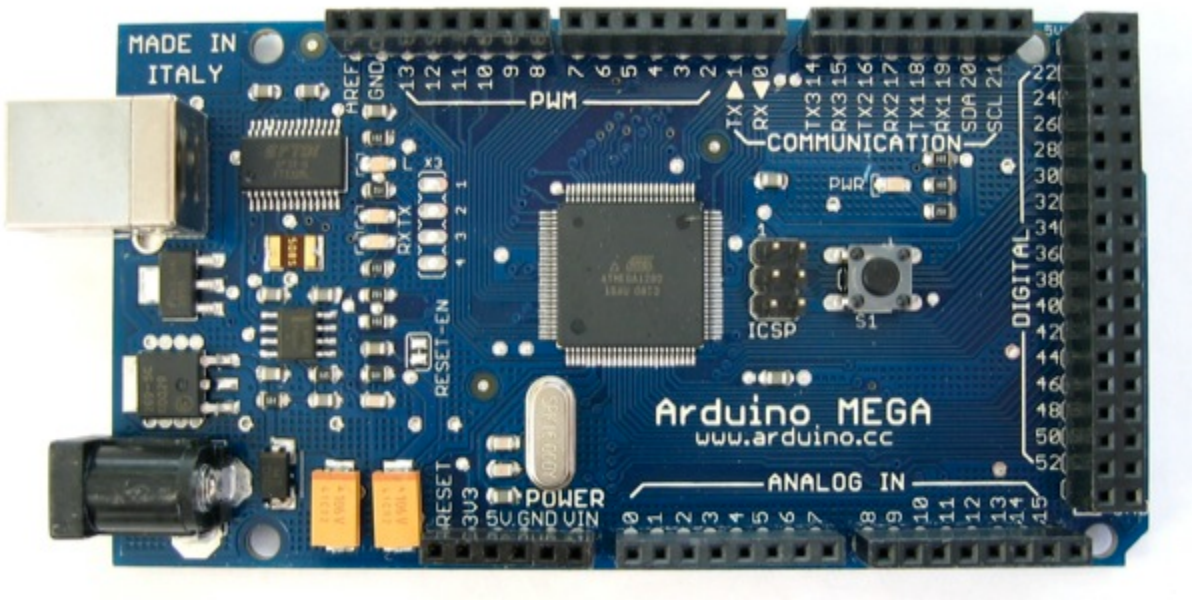
2.1 **Mikrokontrolleri valik**

Arvestades sellega, et autor polnud ennem seda projekti mikrokontrollerite programmeerimise ja nende võimalustega väga hästi kursis, siis eeldas see algselt uurimist ja õppimist. Soovituste ja uurimuse tulemusena sai valituks Arduino mikrokontroller. Testimise käigus sai kasutatud küll mitut erinevat kontrollerit, kuid esmane kontroller, mis kasutusele võeti ning mille põhjal koodid ja skeemid on loodud on Arduino Mega plaadil, millele on integreeritud ATmega2560 mikrokontroller**.** Kuna sellel kontrolleril on rohkem mälu, sisendeid/väljundeid, kiirem taktsagedus ja võimaldab suuremat voolu väljundisse anda, siis sai selle kasuks valitud. See tagab varu ruumi, juhul kui süsteemi arendamiseks ideid tuleb. Lõppkujul kubu realiseerimisel tuleb muidugi arvesse võtta kubu enda mõõtmeid ning selle paigaldamisvõimalusi. Nendest parameetritest lähtudes võib juhtuda, et väiksematele kubudele tuleb kasutusele ka väiksem kontroller, kuid see selgub alles tellimuste käigus.

Arduino põhiliseks eeliseks on kasutajamugavus, mistõttu paljud spetsialistid eelistavad samuti antud kontrollereid. Nimelt on Arduino plaadil kontroller kokku ühendatud juba sisendite ja väljaviikudega, mida on kerge kõikvõimalike aplikatsioonidega ühendada. Samuti on Arduino loonud programmeerimiseks oma keskkonna, kus nad vastavate raamatukogude[1] kasutusel on juba ära defineerinud mikrokontrolleri sisendid ja väljundid plaadil olevatega**.** See jätab kasutajal ära nii-öelda füüsikalisel või kõige algsemal tasandil mikrokontrolleri programmeerimise ning võimaldab neil kiiresti asuda omale vajaliku programmi loomise juurde. Lisaks sellele on nende tarkvaral veel enamgi eeliseid. Näiteks kontrollib see alati üle, kas kasutataval kontrolleril on piisavalt mälu, et programmi talletada ning ei hakka vastasel juhul seda üldse peale laadima. Kõik see teeb antud juhul mikrokontrolleri valiku üpris selgeks ning soovituslikuks nende jaoks, kes antud teemas ennast kõige tugevamini ei tunne. Võib veel lisada, et õpetusi ja kasutajate poolset tagasisidet on nendele meeletus koguses, nii et kindlasti väga hea õppebaas. Peale nende mõningate erandite ning teiste Arduino tarkvara poolsete modifikatsioonide, on programmeerimiskeelena kasutusel siiski C++. See on üheks enim kasutusel olevatest

programmeerimiskeeltest ning peaks seetõttu koodides paljude lugejate jaoks ka mõistetav olema.

Antud Arduino plaati võib toita kas USB kaabli või alternatiivse välise toitega soovituslikult 7-12V piires. Mikrokontrolleril on 256KB flash mälu programmide jooksutamiseks ning 4KB EEPROM mälu, mida saab kasutada andmete salvestamiseks. See võib tunduda üpris väike maht, kuid kui seda nutikalt kasutada, on sellega võimalik päris palju andmeid koguda. Hiljem mõõtmiste ja andurite testimise osas tuleb see ka kastutusse.



Joonis 2.1 Arduino Mega mikrokontrolleri moodul

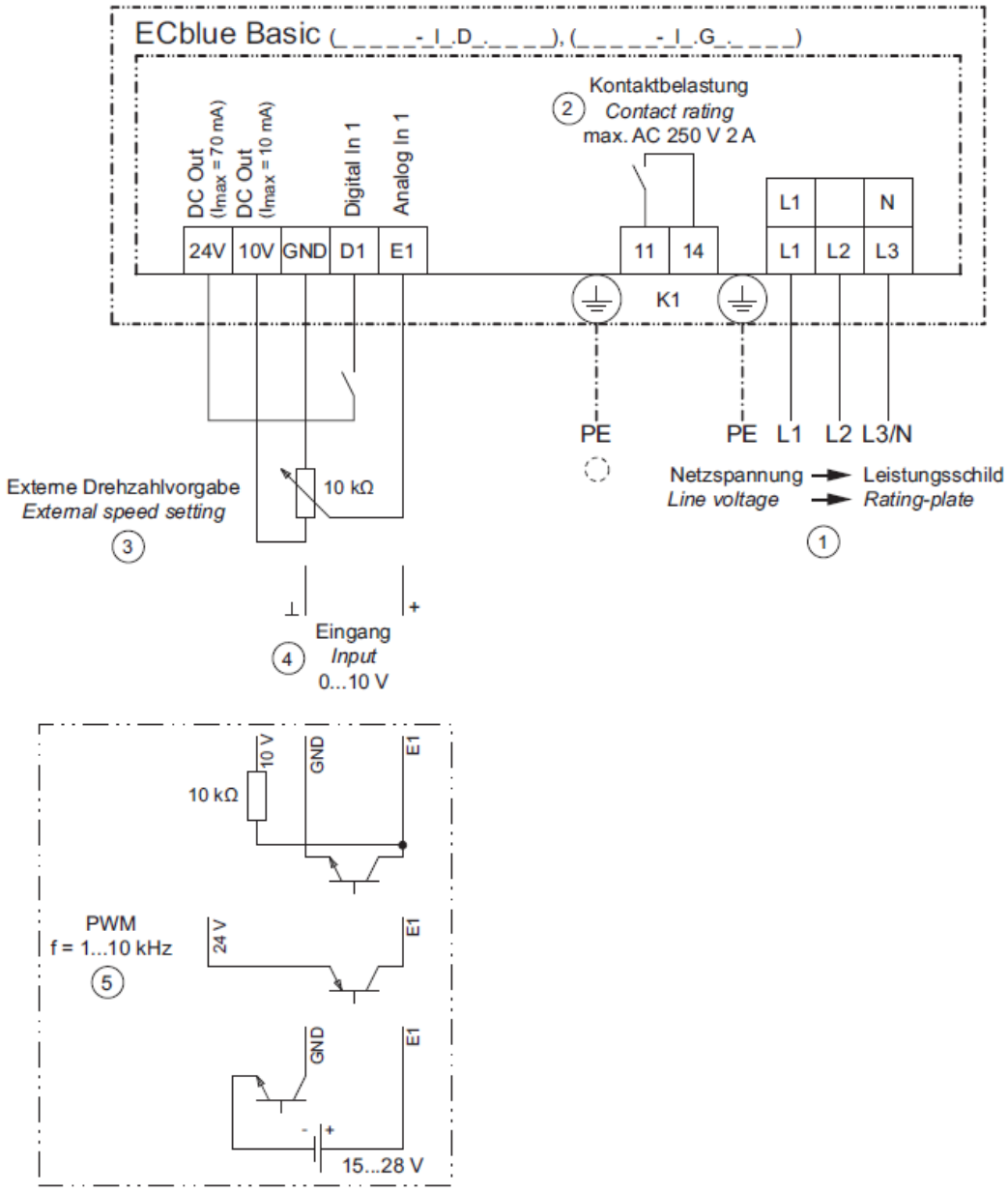
2.2 **Mikrokontrolleri tehnilised andmed:**

MCU3	ATmega2560
Tööpinge	5V
Sisendpinge (soovituslik)	7-12V
Sisend pinge (limiidid)	6-20V
Digitaalsed väljundid/sisendid	54 (millest 15 võimaldavad PWMi)
Analoogsed sisendid	16
Alalisvool väljundi/sisendi kohta	40 mA
Alalisvool 3.3V väljundile	50 mA
Flash4 mälu	256 KB
SRAM	8 KB
EEPROM	4 KB
Taktsagedus	16 MHz

3. Komponendid

3.1 **Ventilaator**

Antud projekti puhul sai kasutatud sama ventilaatorit, mis ettevõte kasutab teiste kubude puhul standardlahenduses. Tegemist on saksa ettevõtte Ziehl-Abegg EC Blue tooterühmaga, kuhu kuuluvad erineva võimsusega ventilaatorid, mida valitakse vastavalt lahenduse vajadusele. Prototüübi puhul sai kasutatu RH35V-ZIK.DC.1R nimelist mudelit. See ventilaator on antud projekti puhul väga hea, kuna sellesse on sisse integreeritud PWMi funktsioon, mistõttu ei ole vaja kasutada sagedusmuundurit ventilaatori kiiruse muutmiseks.
Toote infolehelt saadud ühendus skeemilt (joonis 3.1) on kõige selgemalt näha, kuidas selle ventilaatori juhtimine toimub. Ventilaatori töötamiseks on vaja teha päris mitu ühendust. Esmalt tuleb anda toide standardsest vahelduvvoolu võrgust L1, L2 või L3 sisendisse. Edasi on vaja sulgeda vooluring punktis K1 ehk sisendite 11 ja 14 vahel. Antud joonisel on sinna märgitud ka lüliti, mille rolli selles töös täidab relee. Kõige lõpuks on vaja ka anda 24V signaal digitaalsele sisendile D1 ja 0-10V vahemikus olev signaal analoogsisendisse E1. Vastavalt viimasele pinge väärtusele muutub automaatselt ka ventilaatori kiirus. Selle pinge muutmiseks on joonisel välja toodud 10kOhmi suurune potentsiomeeter, mis on ka antud lahenduses manuaalse juhtimise realiseerimisel kasutusel. All on ka eraldi välja toodud võimalus PWMi kasutusel ventilaatorit juhtida. Seda tehakse läbi n-tüüpi MOSFET[9] transistori, millele vastava sageduse saab genereerida mikrokontrolleri väljundist. Antud ventilaatorisse on sisse ehitatud alaldi ning pingemuundur, mis võimaldavad kasutada kahte kõige vasakpoolsemat väljundit 24V ja 10V pingeallikatena. See lihtsustab antud süsteemi, kuna pole vaja väliseid alaldeid kasutata. Prototüübi testimise käigus sai aga küll kasutatud välist toidet, kuna sellel oli potentsiomeeteriga elektriring kohe sisse ehitatud ning see tegi testimise palju mugavamaks. Samuti on võimalik antud ventilaatorile juurde tellida tootja poolne kasutaja liides, mis on realiseeritud kaugjuhtimispuldil, kuid sellega on võimalik süsteemi juhtida vaid manuaalselt ning seetõttu ei sobi see antud töö jaoks.

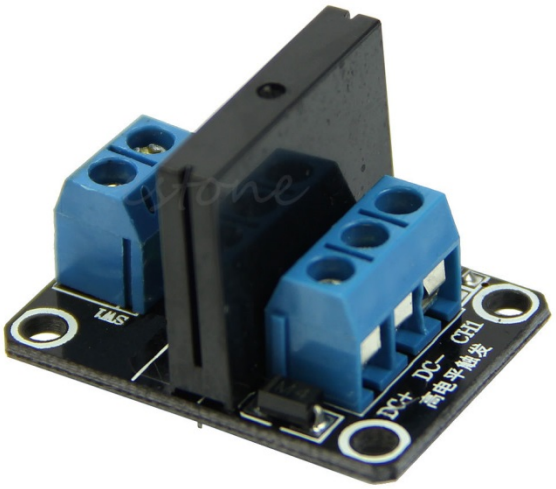


1 *Line voltage ☞ rating plate*
2 *Contact rating max. AC 250 V 2 A*
3 *External speed setting*
4 *Input 0...10 V*
5 *PWM input, f = 1...10 kHz*

Joonis 3.1 EC Blue ventilaatori ühendusskeem

3.2 **Releed**

Algselt sai süsteemi testitud mehaaniliste releedega, mis iseenesest toimis väga hästi, kuid ei olnud nii efektiivsed kui opto-isoleeritud releed. Otsus vahetada tuli puhtalt uurimuslikul teadmisel. Mehaaniliste releede miinuseks on esiteks see, et nad tekitavad ümberlülitamisel müra ning teiseks nad „kuluvad“ kiiremini läbi.

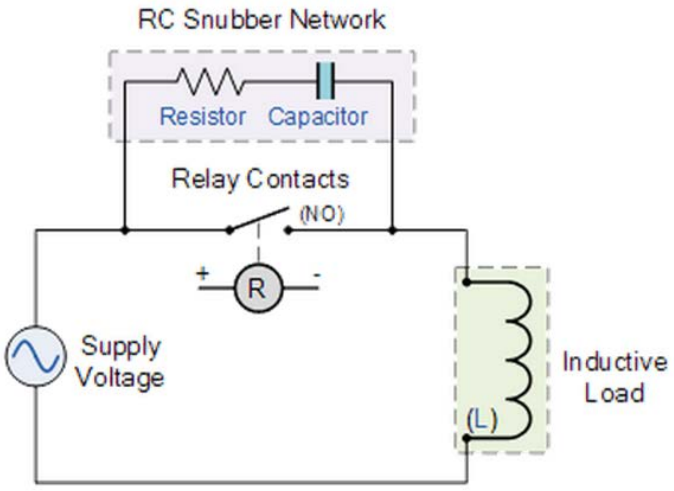


Joonis 3.2 Pooljuhtrelee

Kuna releesid kasutame antud süsteemis vaid komponentide sisse ja välja lülitamiseks, siis see ümberlülituste kord on piisavalt väike, et müra ei ole siin kohal oluliseks faktoriks. Põhjalikumal uurimisel tuli välja, et mehaaniliste releede eluiga ei sõltu ainult nende ümberlülitamiste arvust ning voolu ja pinge suurusest, vaid ka sellest, mis liiki vooluga on tegemist. Vahelduvvoolu puhul ei ole see nii oluline aspekt, kuna seal vool pendeldab pidevalt läbi null punkti.
Alalisvoolu puhul on vool seevastu pidevalt nullist erinev ning iga ümberlülitus põhjustab relee kontaktidel sädeülekannet, mis aja möödumisel kontakti materjali kahjustab ning kui see siis ära oksüdeerub, siis ei juhi ta enam voolu edasi ning relee tuleb välja vahetada.
Kuna opto-isoleeritud releed toimivad valgusülekandel, siis igasugune mehaaniline hõõrdumine ja kulumine jäävad siin kohal ära. Samuti ei põhjusta antud releed vähimatki müra ning nende ümberlülitamise kiirus on oluliselt parem toodud alternatiivile. Eelnevalt kirjeldatud sädeülekanne võib siiski säilida, eriti juhul kui relee lülitab ümber kõrgemal pingel toimivaid alalisvoolu ühendusi või juhib induktiivset koormust. Seetõttu on soovitatav lisada releele

paralleelselt RC ahel (nn Snubber Network ehk sagedusfilter), nagu näha ka joonisel 3.3. Selle ahela ülesandeks on juhtida kõrged pingeimpulsid relee kontaktidest mööda ning seega jäävad need kahjustamata.

Algselt oli plaan pulsi pikkuse modulatsioon (PWM) luua samuti opto-isoleeritud relee baasil, kuid uurimiste tulemusel sai kindlaks, et n-tüüpi väljatransistoril on selle realiseerimine lihtsam ja tõhusam. Kuna uurimise käigus tuli välja, et erinevate pingete ja vooluliikide puhul on parem kasutada eraldi releesid, siis on nelja kanaliga relee asemel kasutusel neli ühe kanaliga releed. See teeb samuti hiljem vahetuse kergemaks, juhul kui mõni relee peaks riknema.

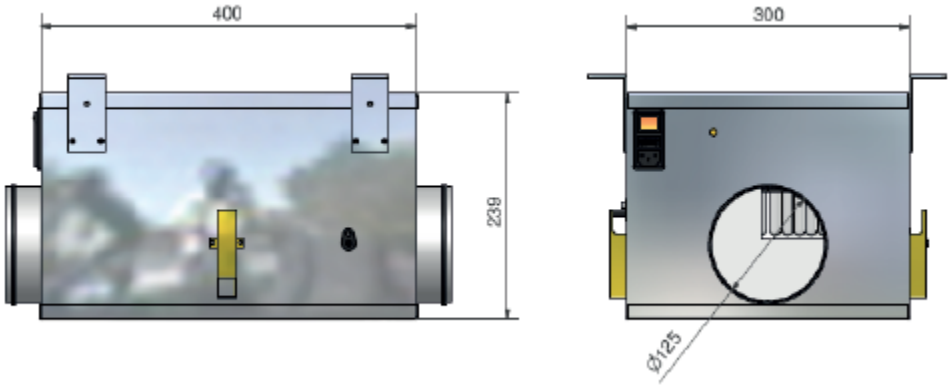


Joonis 3.3 Sagedusfiltriga lisaahela aseskeem

3.3 **Osoonigeneraator**

Osoonigeneraatorist õhu läbi juhtimine on vajalik, kuna kubusse sisse integreeritud filter ei suuda õhku täielikult puhastada. Söögi valmistamisel eristuvad aurud on täis rasvaosakesi ning muid gaase, mis filtreerimata jättes ladestuvad ventilatsioonitorude siseseintele ning tulekahju korral süttib peatamatult terve süsteem. Samuti põhjustavad nad korrosiooni ladestunud

pindadele ning osad gaasid ei pruugi olla keskkonnasõbralikud. Osoon on kahjulik ka inimestele, kuid õhus kiirelt hajuv, seetõttu peab osoonigeneraatori paigutamisega kindlustama, et sellele jääks vähemalt 2-3 meetrit enne kui edasi liigutav õhk võiks kokku puutuda inimestega. Seda, kui palju osooni antud generaator toodab, ei ole võimalik ise reguleerida, kuna see on paika pandud vastavalt keskkonna seadustikule. Seetõttu jääb antud projektis oluliseks vaid relee abil osoonigeneraatori sisse- ja väljalülitamine. Antud generaatori infoleht on ka lisatud kasutatud materjalide alla.



Joonis 3.4 Osoonigeneraatori joonis ja mõõdud

3.4 **Valgustus**

Senini on ettevõte ETS Nord kasutanud oma kubudes luminofoor torusid. Kuna energia säästlikkus oli üks tähtsamaid aspekte antud süsteemi puhul, siis said selle projekti jaoks kõik halogeenlambid vahetatud LED[2] lampide vastu. Kuna LEDid on tänapäeval palju efektiivsemad, siis läheb see vahetus käiku peagi kõikide kubude jaoks. Kuna autor ise oli ka ametis kubu valgustuste välja otsimisega, siis sellest tuleb ka natukene põhjalikumalt juttu.
Valgustuse puhul oli väga mitu parameetrit, mis selle valiku keeruliseks tegid. Tähtis oli see, et valgus oleks suunatud üpris väikese nurga all ning kaks kõrvuti lampi ei põhjustaks liitumiskohti, kus valguse intensiivsus oleks liiga suur ning hakkaks töötajatele silma peegelduma. Üleüldse on optimaalne valguse intensiivsus tööpinna igas punktis ligikaudu 500lx, mis jääb valgustitest

enamus juhtudel umbes 1.5m – 2.0m kaugusele. Enamus lakke uputatavatest LED valgustitest, mis tänapäeval turult leiab, osutusid olema liiga nõrga valgusvooga, et antud vajadust rahuldada. Samuti on tähtis toidu valmistamise puhul see, et valgus ei moonutaks toidu värvi. Seetõttu oli vajalik, et valgusspekter jääks võimalikult lähedale päevavalgusele. Peale uurimist ning testimist sai vastavaks värvi soojuseks valitud 4000K. Kuna lampidele sai ümber tehtud alumiiniumkorpus ning ette pandud kaitseklaasid, siis IP klass ei olnud selle projekti puhul esmatähtis. Peale Osrami spetsialistidega läbi arutamist sai välja valitud nende toode PL-CN50-1100-840-24D-G1, mille kohta on rohkem informatsiooni materjalidesse lisatud infolehel. Valgustusele on võimalik lisada ka dimmer, kuid see standardlahendusse sisse ei lähe, kuna kokkade tagasiside alusel ei olnud antud valgustuse puhul see oluline. Seetõttu on süsteemis valgustuste puhul ainult oluline valgustuse sisse- ja väljalülitamine.

Joonis 3.5 LED valgustusmoodul

3.5 **Summer**

Tulekahju korral ehk siis kui suitsuanduri väärtus ületab kriitilise piiri läheb tööle suitsualarm. See on realiseeritud kasutades piesoelektrilist summerit. Valitud summeril on võimalik muuta helisagedust kasutades PWMi[8] funktsiooni mikrokontrolleril. See võimaldab tekitada vastava sagedusega signaali, mida ülejäänud mürade juures oleks võimalik ära tuvastada. Summerile, nagu igale teisele komponendile, on samuti lisatud manuaalseks väljalülitamiseks eraldi nupp. See on vajalik eelkõige selle jaoks, et kui väikese tulekahju korral summer tööle hakkab, siis ei ole vaja seda häirivat signaali taluda kuni probleem on likvideeritud. Peale normaaltingimuste taastamist saab summeri tagasi sisse lülitada.

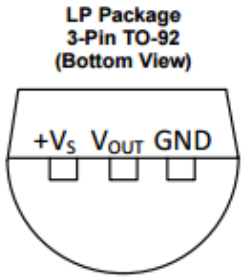
Joonis 3.6 Piesoelektrilise summeri moodul

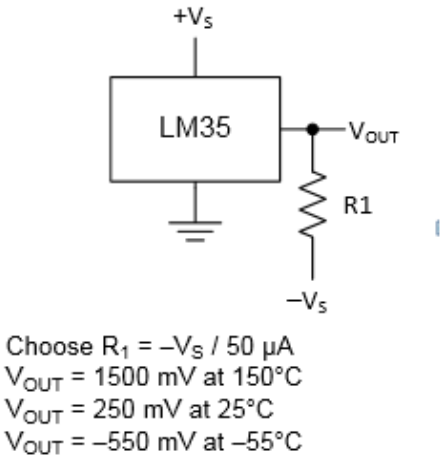
4. Andurid

4.1 Temperatuuriandurid

4.1.1 Temperatuuriandur

Kõigepealt sai testitud LM35 temperatuuriandurit, kuna see oli esimese kohale jõudnud tellimuse sees. Anduri valik tulenes eelkõige juhendaja soovitusest. Samas on tegemist ka ühe enim kasutatava temperatuuri anduriga, kuna ta on väga odav ning teda on väga lihtne väga paljudes valdkondades rakendada. Peale selle kõige tähtsam omadus on tal muidugi see, et ta annab ka mõõtmisel üpris täpsed tulemused. Infolehel on antud mõõtetolerantsiks ±¼-¾°C, mis testimisel ka paika pidas, kuigi nii väikest erinevust on väga raske tõestada ning see sõltub ka teiste testitavate andurite ja termomeetrite täpsusest. Ainukene suurem puudus antud andurid on tema suur reaktsiooniaeg, mis teeb mõõtmise pidevate temperatuuri muutuste puhul keeruliseks. Sellest tingitult jäi antud andur sekundaarfunktsiooni täitma ehk ruumi ja sissepuhke õhu temperatuuri mõõtmiseks.
Antud temperatuuri andur on TO-92 korpuses ning omab kolme sisend/väljund viiku:
+Vs – 4-30V positiivne toite klemm.
GND[5] – 0V ehk maa, alalisvoolu puhul negatiivne toite klemm
Vout – anduri tagasiside klemm, kus väljastatakse väärtus vahemikus -550mV kuni 1500mV.
Läbi tarkvaral põhineva kalibreerimise, saab antud pinge väärtused panna võrduma vastavate temperatuuri väärtustega. Joonisel 4.2 on võimalik näha vastavaid väärtusi anduri ekstreempunktides ning toatemperatuuril töötades. Lisatakisti R1 on vajalik vaid juhul, kui mõõdetakse negatiivseid temperatuure. Selle ärajätmise puhul on mõõtmisvahemik 2 - 150°C.



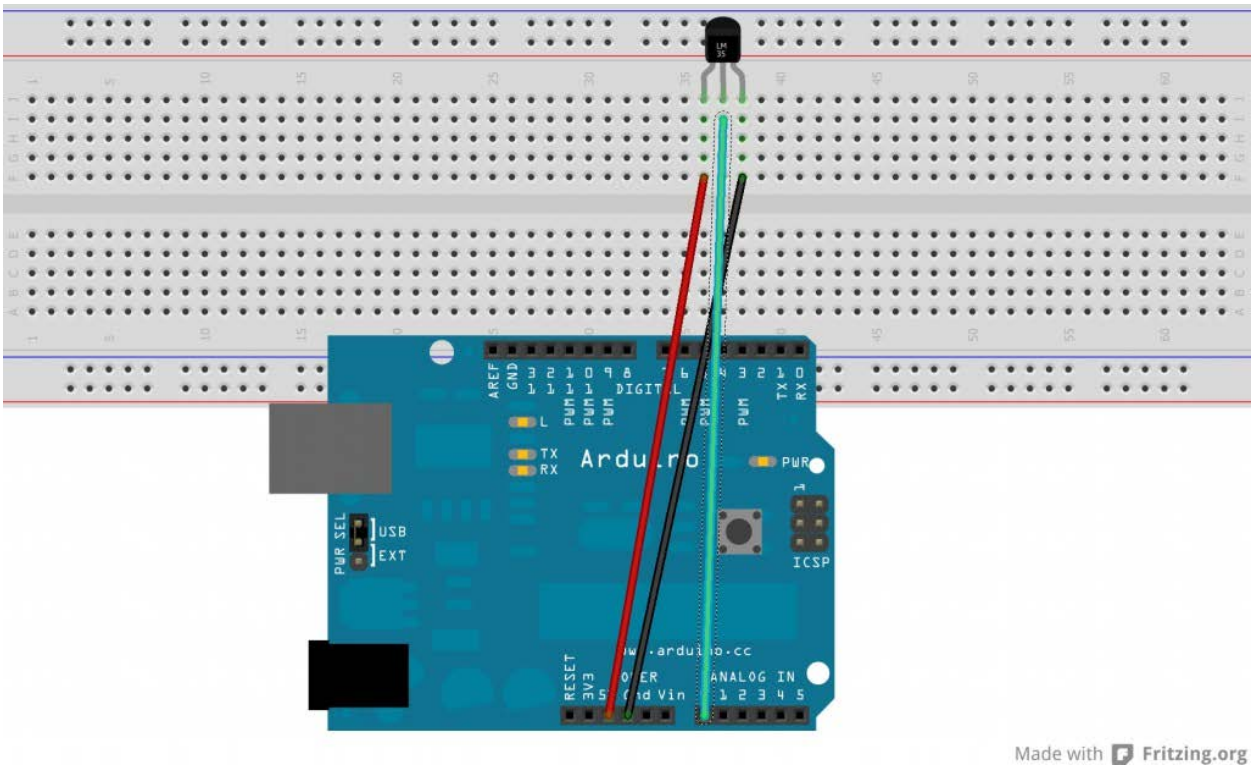


Joonis 4.1 Temperatuurianduri väljundid Joonis 4.2 Temperatuurianduri ühendusskeem

Esmane testimine toimus kõikide anduritega reaalajas. LM35 anduri puhul on selle ühendamine Arduino kontrolleriga väga primitiivne ning realiseeritud järgmiselt:

LM35		Arduino plaat
+Vs	→	5V väljund
GND	→	GND
Vout	→	A0 (esimene analoogsisend)

Joonisel 3.3 on sama ühendus illustratiivselt välja toodud. Analoogselt võiks kasutada ka ükskõik millist teist 15st Arduino Mega analoog sisendist, kuid lihtsuse mõttes on pandud see vastavusse järjekorras esimesega.



Joonis 4.3 Temperatuurianduri ühendusskeem

Tehnilised andmed:

Tundlikkus: 10mV/°C (lineaarne)

Täpsus: ±0.5°C

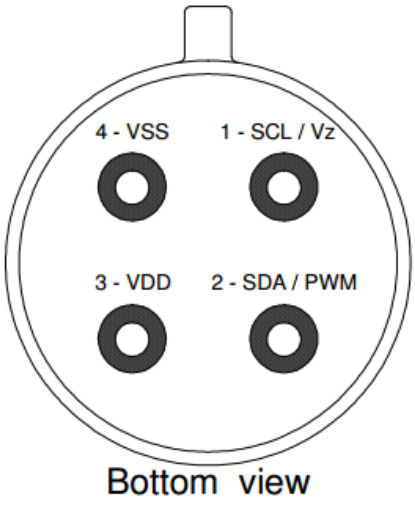
Mõõtmisvahemik: −55°C kuni +150°C

Toitepinge (+VS): 4V-30V

Maksimaalne voolutarve (Imax): 60µA

4.1.2 Infrapuna temperatuuriandur

Infrapuna temperatuuri anduri valik ei olnud nii kerge, kuid seda lihtsustas asjaolu, et enamus antud anduritest on praeguseni väga kõrges hinnaklassis. Selle põhjustavaks faktoriks on ka see, et nende eesmärk on mõõta temperatuuri eemalt ehk nad ei ole antud keskkonnaga otseses kokkupuutes. Kuna antud projektis ei ole tähtis antud lugemite täpsus, siis leidis autor ka mõistliku hinnaga sensorid, mis küll ei ole mõeldud sellise distantsi pealt mõõtma, kuid on siiski siinsesse rakendusse sobivad. See on sellepärast nii, et seda andurit kasutame küll 1.5 – 2.0m kauguselt temperatuuri mõõtmiseks, kuid vastavaid andmeid on vaja vaid süsteemi sisse ja välja lülitamiseks. Seega pliidi sisse lülitamisel võib temperatuur muutuda mitme saja kraadi vahemikus, kuid kui andur tuvastab juba 5°C temperatuuri tõusu, siis on selge, et tööpind on sisse lülitatud.

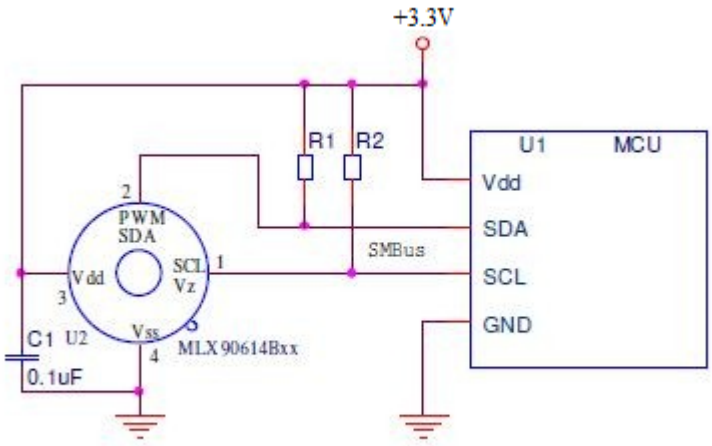


Joonis 4.4 Anduri väljundid

Joonis 4.5 Infrapuna temperatuuriandur

MLX90614ESF-ACF		Arduino plaat
VDD	→	3.3V/5V väljundisse
VSS (Gnd)	→	GND
SDA[6]	→	SDA (Digital pin 20)
SCL[7]	→	SCL (Digital pin 21)

Nagu näha on võimalik antud andurit toita mikrokontrollerist nii 3.3V kui ka 5V väljundpingega. Esijalgu pani autor antud skeemi ka kokku nagu joonisel 5.6 on näidatud. Jooniselt endilt on ka näha, et Vdd suurus on sellise ühenduse puhul 3.3V. See üksinda pole aga piisav, et anda välja õigeid tulemeid, nii et sama väljundpinge on vajalik paralleelselt sisestada ka SDA ja SCL jalgadele, ent voolu peab takistite R1 ja R2 takistama. Vastavate takistite väärtused sõltuvad I²C siin kogumahtuvusest ning on arvutuslikud, kuid antud lahenduses sobib nendeks väärtusteks 4.7kOhmi. Kondensaator C1 on lihtsalt lisatud mürade neutraliseerimiseks, kuid katseliselt sai kindlaks tehtud, et tulemused ei erine selle puudumise korral. Uuringute ja testide tulemusel tuli välja, et andurit on ka võimalik kasutada 5V toitepingega. Peale selle tuli välja, et ainult ühe anduri kasutamise puhul ei ole lisatakistid ning paralleelühendused vajalikud. See avastus tegi töö märksa lihtsamaks, kuid autor soovib mainida, et siin kohal peab väga ettevaatlik olema, sest MLX90614ESF seeria andurite kõik versioonid 5V toitepinget ei võimalda.



Joonis 4.6 Infrapuna temperatuuriandurite ühendusskeem

Testimise käigus selgus, et nende sensorite jaoks on ka loodud Arduino raamatukogu, mis kasutajate tööd märkimisväärselt lihtsustab. Vastavas raamatukogus on temperatuuri lugemid toodud nii Celsiuse kui ka Fahrenheiti skaalas, kuid viimane on autori poolt juhtkoodist likvideeritud, kuna see ei ole seal esimesel lähenemisel vajalik. Lisaks toob see andur välja nii temperatuuri väärtuse anduri enda juures, kui ka vastavas punktis, kuhu infrapuna kiirgus on

suunatud, mis teeb sellest universaalse lahenduse. Siin kohal peab muidugi lisama, et andur võtab keskmise väärtuse temperatuuridest 90 kraadise nurga all. See tähendab, et täpset mõõtmist suuremate kauguste puhul ei ole mõtet oodata, kui ei ole juurde ehitatud väga täpset läätsede- või peeglitesüsteemi infrapuna kiirguse suunamiseks. Kuna antud projektis ei ole aga vastava temperatuuri väärtuse täpsus niivõrd oluline, siis ei ole edasist füüsilist modifitseerimist siin kohal vaja teha. Algselt oli süsteemis mõeldud ka LM35 andur panna väljatõmbe õhu temperatuur mõõtma, kuid katsete tulemusel oli näha, et vastav infrapuna andur annab sama usaldusväärseid tulemusi ning selle reageerimisaeg on tunduvalt parem eelnevalt nimetatust. Seetõttu sai ka see andur valitud juhtima ventilaatori kiirust.

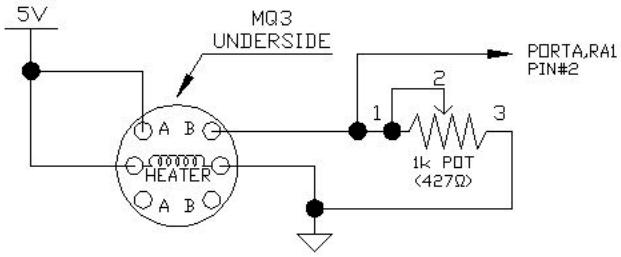
Tehnilised andmed:

Täpsus: ±0.5°C
Mõõtmisvahemik: −40°C kuni +125°C (anduri ümbruses)
–70°C kuni +380°C (objekti temperatuur)
Toitepinge (+VS): 3.3V või 5V
Kiiretoimelisus (võrreldes LM35 anduriga)

4.2 **Suitsuandurid**

Suitsuandurina on antud projektis kasutusel MQ3 gaasiandur. Tegelikult sai tellitud kaks erinevat gaasiandurit, nii MQ2 kui ka MQ3, mis mõlemad tuvastasid teiste gaaside seas ka suitsu. Katsete tulemusena aga sai nende seast välja valitud viimane. See oli eelkõige selle tõttu, et antud anduri tundlikkus keskkonna parameetri muutusele oli parem. Samade katsete puhul oli MQ3 anduri lugemi väärtuste intervall suurem. Kuna algselt oli plaanis kasutada seda andurit ka ventilaatori kiiruse muutmisel, siis sellepärast sai ka parema anduri kasuks otsustatud. Samas on tegelikult antud anduri ülesanne vaid tulekahju alarm tööle panna, kui anduri lugem ületab kriitilise piiri. Seetõttu oleks siinses kasutuses toiminud väga hästi ka MQ2 gaasiandur. Samas erinesid vastavad andurid ka teineteisest selle poolest, et nende tundlikkus oli parem erinevate gaaside

suhtes. MQ2 tunneb õhust paremini ära metaani, butaani, **LPG** ning MQ3 on etem alkoholi ja etanooli jaoks. Kuna osades söögikohtades on kasutusel gaasipliidid ning nendest erituvad gaasid võivad MQ2 anduri puhul väga kõrgeid tulemeid anda, siis oli ka see üks põhjuseid, miks teise anduri kasuks sai otsustatud. Lisaks sellele sai MQ3 valitud koos välise plaadiga, millel oli potentsiomeeter peal, nagu on ka näha joonisel 4.7. See võimaldas paremini lugemite vahemikku koodiga kalibreerida.



Joonis 4.7 Gaasianduri ühendusskeem

Joonis 4.8 MQ3 gaasianduri moodul

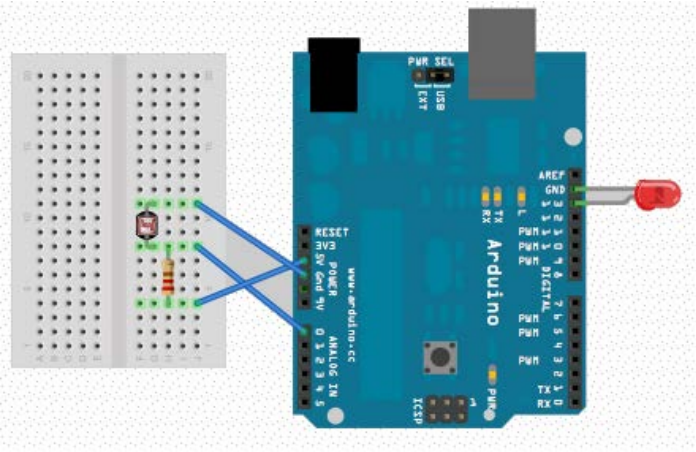
MQ3		Arduino plaat
+Vs	→	5V väljund
GND	→	GND
Vout	→	A1 (teine analoogsisend)

Antud gaasianduril on tegelikult olemas nii digitaal- kui ka analoogväljundid, kuid antud süsteemis sai lihtsuse mõttes valitud ainult üks neist, milleks oli analoogväljund. Iseenesest oleks saanud samahästi ka kasutada digitaalset väljundit, kuid kuna fototakisti ja LM35 temperatuuri andur olid samuti analoogväljundiga ühendatud, siis sai ühtluse mõttes ka suitsuandur samamoodi ühendatud.

4.3 **Fotoresistor**

Fototakisti ülesandeks antud süsteemis on ainult aru saada, kas töökeskkonnas on valgustus sisselülitatud või mitte. See funktsionaalsus on loodud energia säästmiseks. Kui kubu all töötamine on lõpetatud, siis süsteem toimib veel niikaua, kuni temperatuuri ja gaasiandurite näidud on üle väljalülitamiseks sobitatud piiri. Peale seda, kui aga süsteem on välja lülitatud, siis mikrokontroller töötab endiselt edasi. Selleks, et mikrokontroller panna säästurežiimi[10] kasutamegi antud fotoresistorit. Nagu näha joonisel 4.11, on katsed läbiviidud erinevate valgusintensiivsuste puhul. Täpne kalibreerimine sõltub otsestelt sellest, milline on valgustatavus vastavas töökeskkonnas, kuid üldiselt võib järeldada, et kui lugemi näit jääb umbes 900 ühiku juurde, siis on tegemist töövälise ajaga ning mikrokontroller on ooteolekus ning kui lugemi tulemus muutub madalamaks, siis lülitab MCU ennast sisse ning on valmis süsteemi alustama, kui tuleb vastav tagasiside temperatuuriandurilt. Siin kohal võib veel rõhutada, et temperatuuri anduri näit on hierarhiliselt kõrgemal tasemel fototakisti omast ning seetõttu ei lülita mikrokontroller ennast ooterežiimi enne, kui selleks on juhtalgoritmi poolne luba antud ning süsteemi komponendid välja lülitatud. See on tähtis näiteks sel juhul, kui pliit on ööseks tööle pandud ning tuled ära kustutatud.

Joonis 4.9 Fototakisti



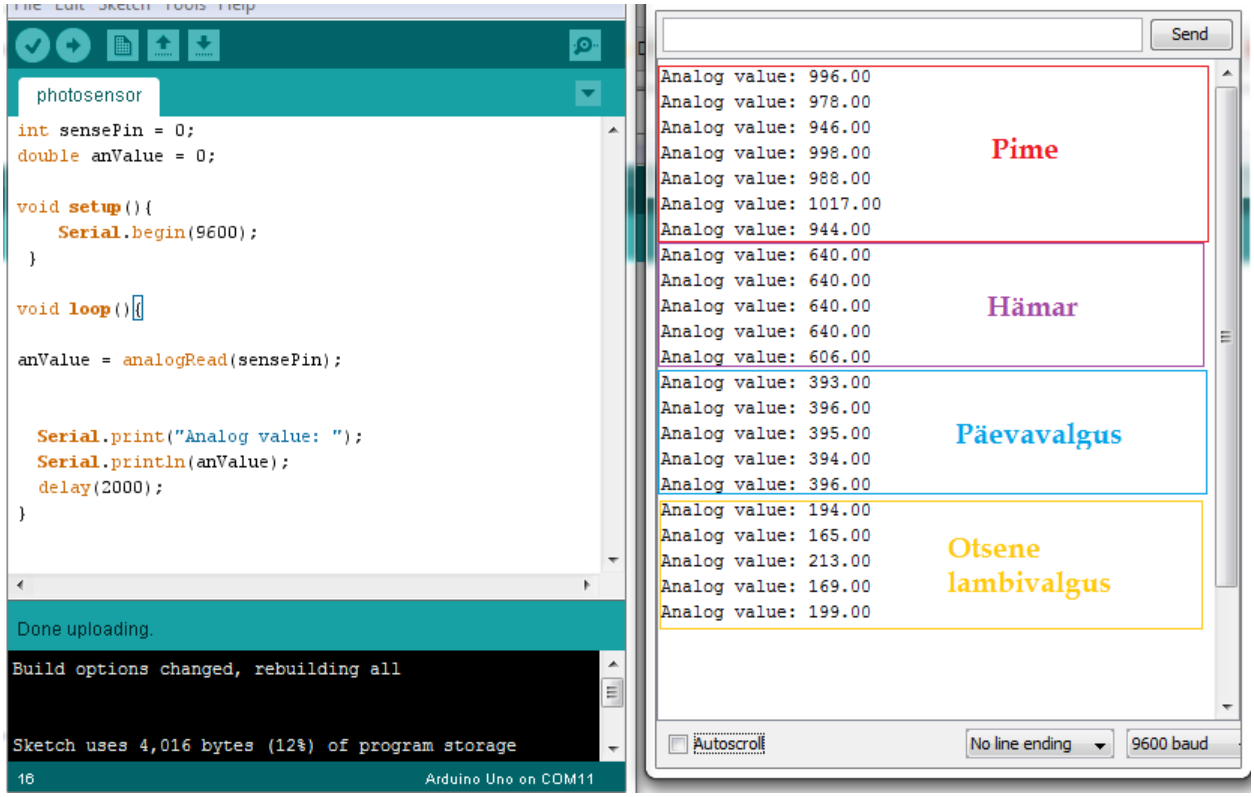
Joonis 4.10 Fototakisti ühendusskeem

GL5528 Arduino plaat

+Vs → 5V väljund (lisaks 10kΩ takisti)

GND → GND

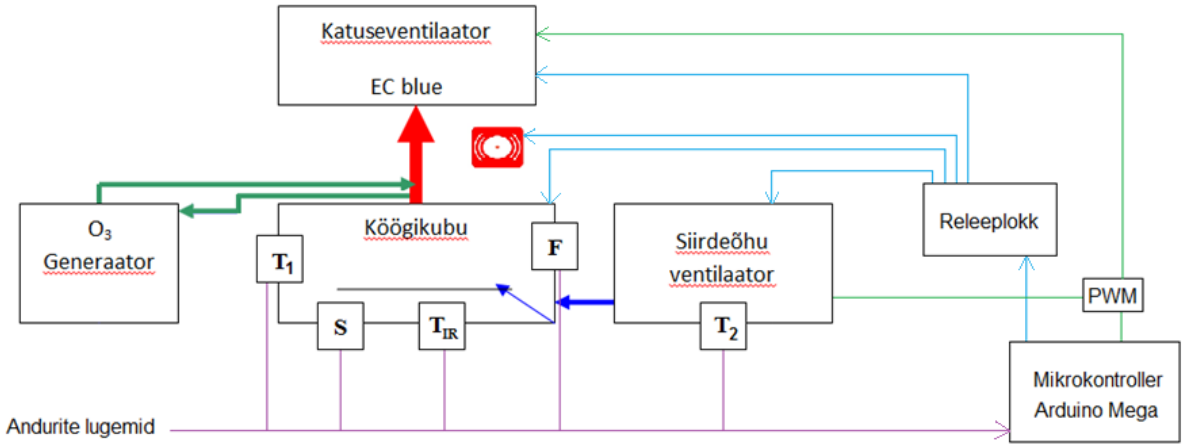
Vout → A2 (kolmas analoogsisend)



Joonis 4.11 Fototakisti mõõtmistulemused

5. Projekti skeemi väljatöötamine

Struktuurskeemil (joonis 5.1) on välja toodud süsteemi komponendid ning nende funktsionaalsus. Vastavate andurite lugemid jõuavad mikrokontrollerisse, mille alusel saab süsteemi komponente juhtida. Releeplokiga on võimalik vastavaid komponente ümber lülitada ning PWMi funktsionaalsust teostab n-tüüpi väljatransistor, millele antava signaaliga mikrokontrolleri poolt saab reguleerida ventilaatori töökiirust.



Joonis 5.1 Struktuurskeem

T1 – Köögikubu välisküljel asuv LM35 temperatuuri andur töökeskkonna temperatuuri mõõtmiseks.

T2 – Sissepuhke õhu temperatuuri mõõtmiseks kasutatav LM35 temperatuuri andur. See on lisavõimalus, kuna enamus köökides, kuhu antud süsteem rakendatakse, ühendatakse sissepuhe ülejäänud hoone ventilatsioonisüsteemiga. Sel juhul ei ole siirdeõhu ventilaatori kiiruse muutmine anduri T2 näidu järgi võimalik.

T_{IR} – Infrapuna temperatuuriandur MLX90614ESF-ACF tööpinna temperatuuri mõõtmiseks ning süsteemi juhtimiseks.

F – Fototakisti GL5528 mikrokontrolleri säästurežiimi lülitamiseks.

- Tulekahju signaalina kasutatav summer.

6. Süsteemi juhtimine

6.1 **Manuaalne juhtimine**

- Võimaldab reguleerida ventilaatori kiirust vastavalt oma soovile.
- Kõikidele komponentidele on lisatud lülitid, juhul kui neid on vaja välja lülitada. Näiteks võib lülitada välja osoonigeneraatori, kui on tegemist vaid ruumi tuulutamisega.

Manuaaljuhtimise sisse lülitamiseks on lisatud lüliti, mida saab täpsemalt näha lisade all olevalt elektriskeemilt. See lülitab välja PWMi funktsiooni täitva väljatransistori ning vool liigub läbi 10kΩ suuruse potentsiomeetri, mille abil on võimalik muuta pinget 0-10V vahemikus, mis omakorda muudabki ventilaatori kiirust. Selline lineaarne kiiruse muutmise võimalus on etem kui enamus süsteemides, kus on võimalik vahetada kiirust vaid kolme või enama kindlaks määratud väärtuse vahel.

NB! Isegi manuaalse juhtimise aktiveerimise puhul jääb alles automaatne funktsioon süsteemi sisse- ja välja lülitamiseks kui kubu all tööd alustatakse või lõpetatakse, juhul kui vastavaid komponente pole manuaalselt välja lülitatud.

6.2 **Automatiseeritud juhtimine**

Tegelikult toimibki süsteem pidevas automaatrežiimis, kui seda pole eraldi välja lülitatud. Automaatse süsteemi puhul piisab töötajatel lihtsalt pliit sisse lülitada ning olenevalt sellest, kui kiiresti jõuab pliit ennast üles soojendada, lülitub ka kogu süsteem sisse. Kuna ventilaator kiirus on andurite näitudega lineaarses sõltuvuses ning viimased erinevad töökeskkondade vahel, siis ei ole võimalik täpseid vasteid anda, kuid ülevaatlikult saab seda teha. Täpne süsteemi kalibreerimine käib vastavalt töökeskkonnale.

Süsteemi tööpõhimõte erinevatel režiimidel:

A. Kubu all ei valmistata toitu, pliit ja valgustus on välja lülitatud.
   T_{IR} – anduri temperatuur = T_1 anduri näit ehk ruumi temperatuur (ligilähedane)
   S – andur ei näita suitsu
   Kõik komponendid on välja lülitatud.

B. Kubu all ei valmistata toitu, pliit on sisse lülitamata, kuid valgustus on sisse lülitatud.
   T_{IR} – anduri temperatuur = T_1 anduri näit ehk ruumi temperatuur (ligilähedane)
   S – andur ei näita suitsu
   Kõik komponendid peale mikrokontrolleri on välja lülitatud.

C. Pliit on sisse lülitatud, kuid toidu valmistamise koormus on väike
   T_{IR} – anduri temperatuur > T_1 anduri temperatuur + 5….10 ^{o}C
   S – anduri näit ei ole üle piirväärtuse
   Ventilaator töötab 10%-50% koormusel
   Komponendid sisse lülitatud.

D. Kubu all käib aktiivne toidu valmistamine
   T_{IR} – anduri temperatuur > T_1 ruumi temp + 10… ^{o}C
   S – anduri näit ei ole üle piirväärtuse
   Ventilaator töötab 50%-100% koormusel
   Komponendid sisse lülitatud.

E. Kubu all on tulekahju
   T_{IR} – anduri temperatuur >> T_1 ruumi temp + 5 ^{o}C
   S – andur näit ületab piirväärtuse.
   Ventilaator ja osoonigeneraator lülituvad välja
   Suitsualarm lülitub sisse.

7. Mõõtmisalgoritmi väljatöötamine

Kuna iga töökeskkond on paratamatult teistest erinev, siis ei ole võimalik luua süsteemi, mis toimiks kõikides paikades samamoodi. Seetõttu on oluline enne süsteemi käima panemist teha ära keskkonna parameetrite mõõtmised. Selleks sai testitud erineva pikkuse ja sisuga programme ning lõppkokkuvõttes jõudis autor järelduseni, et ühe tööpäeva pikkune programm peaks olema piisav, et koguda piisaval hulgal informatsiooni süsteemi kalibreerimiseks. Söögikohtades, kus toidu valmistamine toimub ka öösel, on loomulikult mõistlik pikendada programmi vastavalt tööaegadele.

Vastava programmi loomisega tuli ette ka ette mõningaid anomaaliaid. Algselt sai tehtud kogu programm ühe funktsioonina, kuid see läks natukene liiga pikaks ning siis sai see ümber tehtud alamprogrammide abil, mis tegid koodi umbes kolm korda lühemaks. Hoopis suurem probleem seinsens aga selles, et Arduino mikrokontrolleril on väga vähe kasutatavat mälu, mistõttu oli oluline, et seda saaks ära kasutatud võimalikult efektiivselt. Sooviks oli saada ühe mõõtmisega kätte kõikide andurite optimaalsed mõõtmisvahemikud, kuid seda on väga keeruline 4KB mäluga salvestada. Antud mälumahtu on võimalik kasutada 255 mäluaadressiga, milles iga salvestatud väärtus saab olla samuti maksimaalselt 255 ühikut suur. Seetõttu ei olnud võimalik sinna salvestada andurite näite, mis ületasid vastavat piiri ning oli vaja mõelda välja teistsugune viis tulemuste salvestamiseks.

Nõndaks sai jagatud vastavad mäluaadressid erinevate temperatuuri väärtuste salvestamiseks. Reaalne temperatuuride vahemik, mis vastavates keskkondades võib esile tulla sai valitud 20-61°C. See oli samuti maksimaalne vahemik, mille puhul sai teha kolme erineva lugemiga temperatuuri mõõtmised ning lisaks ka salvestada maksimaalne gaasianduri lugemi näit. Järgnevalt on selgitatud ära mõõtmiskoodi tööfunktsioon, kuid täpsemalt saab selle koodiga tutvuda selle töö lisasektsioonis.

Kõigepealt mõõdetakse tulemused kõikidelt anduritelt ning need salvestatakse ajutiste muutujate alla. Kuna tegemist on ajutiste muutujatega, siis nad kaotavad oma väärtuse programmi lõppedes ning seetõttu ongi oluline nende ümber salvestamine. Esimesse mäluaadressi sai salvestatud temperatuuri väärtused, mis jäid alla 20°C ning iga järgnev aadress sai pandud vastavusse ühe kraadise vahemikuga, mis oli eelnevast suurem. See tähendab, et teise mäluaadressi alla salvestati kõik juhud, kui temperatuur jäi 20 ja 21°C vahele jne. Seega sai kasutatud

mäluaadresse nagu loendureid: iga kord kui vastavas vahemikus olev näit tuli anduri lugemile, sai vastava mäluaadressi väärtust tõstetud ühe võrra. Kuna iga paari sekundi tagant loetakse peale uued andurite näidud, siis on selge, et 255 on liiga väike arv, et terve tööpäeva tulemusi salvestada. Sellepärast tuli kasutuse võtta iga väärtuse puhul kaks loendurit. Peale seda, kui esimene loendur oli jõudnud täis ehk saanud väärtuseks 255, siis järgmise sobiva tulemuse puhul sai selle mäluaadressi väärtus pandud uuest võrdseks nulliga ning võeti kasutusele järgmine vaba mäluaadress, mille väärtus tõsteti ühe võrra. Nõnda on võimalik salvestada 255^2 ehk 65 025 korda iga temperatuurivahemiku jaoks. See on piisavalt suur arv, et rahuldada eelnevalt kirjeldatud nõudeid. Nõndaks jooksebki programm läbi samaaegselt kolm erinevat temperatuuriandurite lugemit ning salvestab nad vastavatesse mälupesadesse. Esimesed 85 pesa katavad infrapuna temperatuuri anduri ümbrise temperatuuri, järgised 85 pliidi pinnale suunatud kiirgusvoo temperatuuri ning viimased 85 jäävad ruumi temperatuuri mõõtmiseks.
Antud süsteem kattis ära täpselt kõik 255 mäluaadressi, kuid lisaks temperatuuri mõõtmistele oli tarvis teada saada ka maksimaalne gaasianduri väärtus, milleni antud andur tööpäeva jooksul jõuda võib. Kõik kolm mõõtevahemikku olid aga samad, kuna selle muutmine oleks koodi veel palju pikemaks ja keerulisemaks teinud. Reaalsetes tingimustes ei tohiks aga ruumi temperatuur kunagi jõuda 60°C juurde ning seetõttu sai kõige viimased mäluaadressid võtta kasutusele selleks, et suitsuanduri maksimaalne tulemus sinna salvestada. Siingi pidi probleemile lähenema väikese ringiga, sest antud anduri väärtused varieerusid 0-1023 vahemikus. Seega sai täidetud viimased neli mälupesa sarnaselt eelneva koodiga, kuid kuna oli vaja teada vaid ühte kindlat väärtust, mitte mitu korda see esineb, siis sai antud väärtus jagatud nelja mälupesa vahel. Juhul kui väärtus ületas 255 piiri, siis sai 252. mälupesa täielikult täidetud ning lugemi väärtusest lahutatud 255. Kui see oli ikka suurem kui 255, siis täideti ka järgnev mälupesa täielikult ning korrati sama protsessi kuni jõuti välja viimasesse mälupessa. Hiljem sai teise koodiga, mis antud mälupesade sisu nö dekodeerib, kõik need tulemused uuesti kokku liidetud ning vastav tulemus ongi täpselt teada. Iseenesest oleks saanud ka kasutada ühte mälupesa ning tulemused lihtsalt kas nelja või kümnega läbi jagada ja hiljem korrutada, kuid see oleks tekitanud ebatäpsuseid, mis oleks võinud tekitada edasisis probleeme ning autor ei tahtnud sellega riskida. Kuna mõõtmiskood ise oli väga mahukas, siis sellega saab lähemalt tutvuda lisasektsioonis.

8. Juhtsüsteemi loomine

8.1 **Süsteemi lühiülevaade**

Juhtsüsteemi käima panemiseks on vajalik eelnevalt keskkonnaparameetrid mõõta eelnevas punktis nimetatud programmi abil. Vastavad temperatuurivahemikud ning teiste andurite mõõtevahemikud ei ole ühtselt defineeritud, kuid antud dokumentatsioonis on kajastatud ühe 8h pikkuse mõõtmise tulemeid. Kõige olulisem tagasiside oli see infrapuna temperatuurianduri näit (T_{amb}), mis mõõtis kubust väljatõmmatava õhu temperatuuri. See vahemik jäi 24°C ja 58°C vahele. Vastavad väärtused sai ümber arvutada 0-255 byte süsteemi, mille jaoks sai loodud järgmine valem:

$$PWM_{value} = 7.5 \cdot (T_{amb} - 24) \quad (8.1)$$

PWM_{value} – vastav väärtus 0-255 vahemikus, millega määrab mikrokontroller läbi väljatransistori ventilaatori kiirust. Sõltuvus temperatuuri lugemitega on välja toodud ka tabelis 8.2.
Vastav samm sai PWMile määratud Microsoft Exceli abil, kus kahe ekstreemväärtuse ning massiivi pikkuse alusel sai vaheväärtused interpoleeritud. Algne väärtus on aga pandud 5 peale, kuna see tagab ventilaatori töölepaneku ilma liigse jõumomendita. Järgnev väärtus antakse juba esimeste mõõtetulemuste alusel, kui süsteem pole lülitatud ümber manuaalsesse juhtimisse.
Kui pliit lülitatakse sisse ning sellele suunatud infrapuna anduri näit ületab ruumi temperatuuri 5^{o}C võrra, siis lülitatakse sisse valgustus ja ventilaator. Osoonigeneraatori sisselülitamisega oodatakse, kuni suitsuanduri näit jõuab üle sellele eelnevalt paika pandud väärtuse. See tagab samuti energiakokkuhoiu nendel hetkedel, kui kubu all reaalset tööd ei tehta, aga küttekehad veel kuumana säilivad.

Nagu eelnevalt oli kirjeldatud, siis osoonigeneraator lülitatakse sisse erinevalt teistest komponentidest. Samuti lülitatakse see vahepeal välja, kui suitsuanduri väärtus langeb vastavalt paika pandud väärtusest väiksemaks. Seda kontrollib programm samuti iga tsükli jooksul.

Andurite mõõtmise jaoks on loodud alamprogramm measure(), kuna seda kutsutakse juhttsüklis välja enam kui ühe korra ning seega on võimalik koodi mahtu vähendada. See alamprogramm küll ei tagasta ühtegi väärtust, kuid vastavad mõõtmistulemused pannakse salvestatakse globaalmuutujate alla ning neid saab seega juhtprogrammis kasutada. LM35 sensori puhul on ka tarvis sisendväärtuse teisendamine Celsiuse skaalale, mille saab siis, kui anduri tulem korrutada läbi konstandiga 0.48828125.

Temperatuuri lugem (*C)	PWM
24	0
25	7.5
26	15
27	22.5
28	30
29	37.5
30	45
31	52.5
32	60
33	67.5
34	75
35	82.5
36	90
37	97.5
38	105
39	112.5
40	120
41	127.5
42	135
43	142.5
44	150
45	157.5
46	165
47	172.5
48	180
49	187.5
50	195
51	202.5
52	210
53	217.5
54	225
55	232.5
56	240
57	247.5
58	255

Tabel 8.2 Temperatuuri näitude vastavus PWMi funktsioonile

8.2 **Tulekahjuprogramm**

Mõõtmistulemustena on veel paika pandud maksimaalsed suitsuanduri ning temperatuurianduri näidud, mida kasutatakse tulekahju programmis, kui vastavate andurite väärtused nendest kõrgemateks osutuvad. Antud mõõtetulemuste juures jäid need väärtused gaasianduri puhul 800 juurde ja temperatuuri anduri näit 60ni. Need on vastavad suurused, milleni testimise käigus ei jõutud, kuid tulekahju korral kindlasti esinevad. See on ka esimene funktsioon, mida juhtsüsteem kontrollib, peale kõikide andurite mõõtetulemuste saamist. Kui süsteem läheb tulekahju programmile, siis lülitatakse automaatselt välja ventilaator ja osoonigeneraator, et vältida tule levimist ventilatsioonisüsteemis. Samuti lülitatakse sisse tulekahju signaal, et anda töötajatele ohust märku. Olles tulekahjufunktsioonis kontrollib süsteem iga sekundi tagant uuesti vastava temperatuurianduri ning suitsuanduri väärtusi. Normaaltingimuste taastumisel lülitatakse tulekahju alarm välja ning ülejäänud komponendid pannakse tööle järgmise juhtsüsteemi tsükli algul, peale uute mõõtetulemuste saamist.

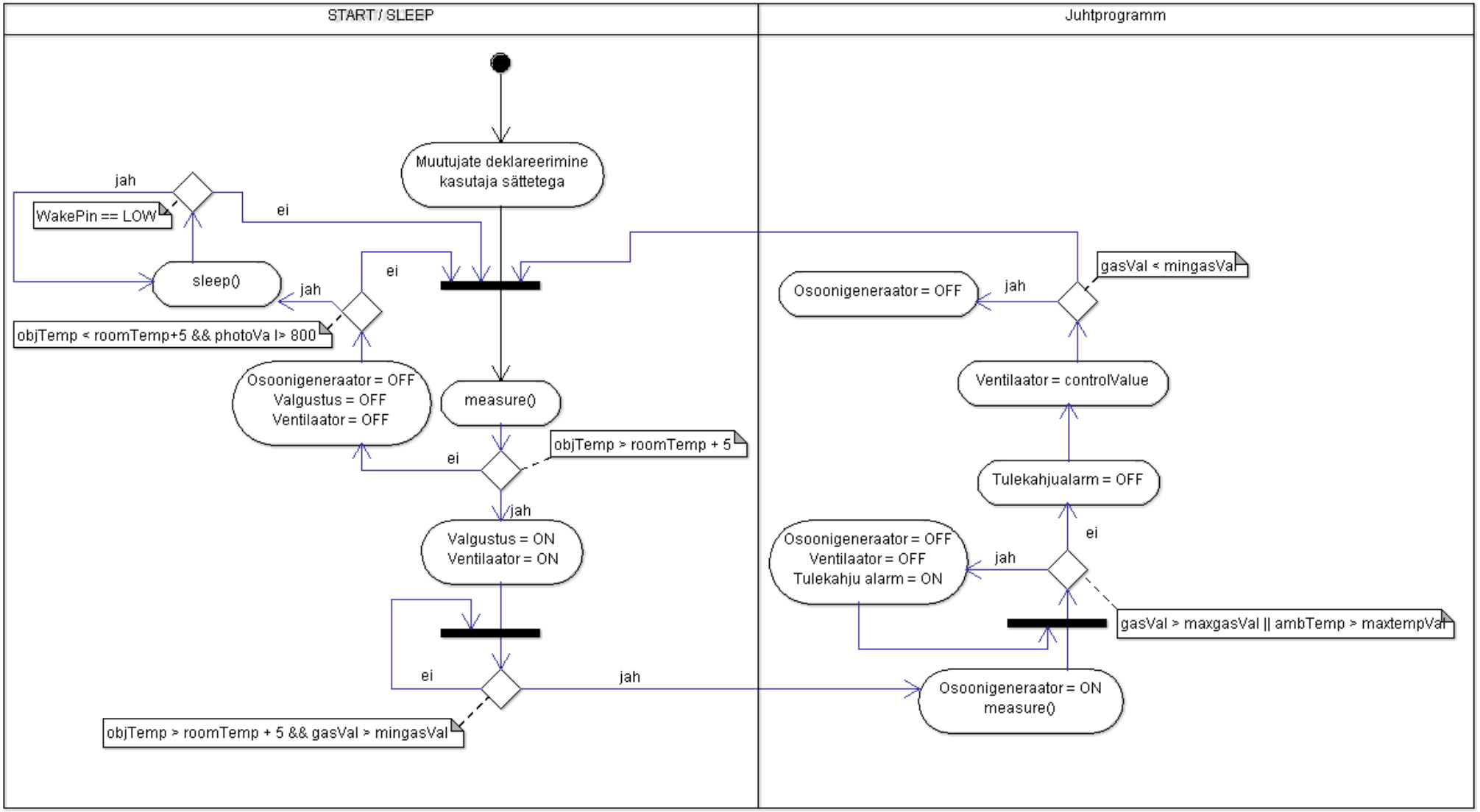
8.3 **Energiasäästurežiim**

Muidu ei ole mikrokontrolleri energiakulu väga drastiline, kuid juhtprogrammi täites kogub ta andurite näitusi üpris tihti ning seega tekiks mõttetu energiakulu, kui töökeskkonda parasjagu ei kasutata. Seetõttu on süsteemi lisatud programm, mis lülitab mikrokontrolleri selleks ajaks säästurežiimi. Selline võimalus on ATmega2560 ja ka paljudel teistel Arduino mikrokontrolleritel sisse ehitatud, kuid selle realiseerimiseks on vaja kasutada vastavaid tootja poolt pakutud raamatukogusid. Tegelikult on olemas 5 erinevat säästurežiimi, kuid antud projekti jaoks on välja valitud kõige energiasäästlikum, milleks on SLEEP_MODE_PWR_DOWN. Olles antud režiimis on küll mikrokontrolleril kõige vähem funktsionaalsust, kuid siin puhul ei ole see ka oluline. Ainukene oluline aspekt on see, et oleks võimalik MCU sisse lülitada vajalikul hetkel.

Antud režiim on lisatud süsteemi alamprogrammina sleep(), mis on koodis ka täpselt lahti seletatud ning seega piirdume siin juhtsüsteemi kirjeldamisega.
Juhtprogramm ei lülita end ennem vastavasse režiimi, kui pliidile suunatud temperatuurianduri näit saab ligilähedaseks ruumi temperatuuriga ning gaasianduri näit langeb alla madalama piirväärtuse. Siin kohal tuleb kasutusele fototakisti, mis mõõdab töökeskkonna valgustust. Kui köögis tuled kustutatakse ja eelnevalt kirjeldatud nõuded on täidetud, saab mikrokontroller märku sellest, et tööpäev on lõppenud ning kutsutaksegi välja vastav alamprogramm, mis kontrolleri säästurežiimi lülitab. Selles olekus ei kogu kontroller mingeid andmeid ega jooksuta ka koodi vaid on lihtsalt ooteolekus. Nimelt ootab ta vastavat signaali, et uuesti käivituda. Antud juhul on süsteem loodud sedamoodi, et andes kõrge[11] signaali teise digitaalsisendisse lülitab mikrokontroller uuesti sisse. Kuna fototakisti väärtus väheneb valgusintensiivsuse tõstmisega, siis seetõttu lähebki süsteem uuesti käima, kui töökeskkonna valgustus sisse lülitatakse. Seejärel hakkab kontroller uuesti anduritelt informatsiooni koguma, et vajadusel taaskäivitada juhtprogramm.

8.4 **Juhtalgoritmi väljatöötlemine**

Juhtalgoritmi selgitus:
Antud programmis ei ole defineeritud lõppu, kuna seda funktsiooni täidab vastav säästurežiim, mida algoritmis on tähistatud tema alamfunktsiooni nimetusega sleep(). Vastavate komponentide olekud on kergemalt haaramiseks tähistatud nende inglise keelsete vastanditega ON ja OFF.
Väidetes (statement) ehk nelinurksetes algoritmi punktides on kasutatud vastavaid tähistusi:
|| - loogiline VÕI tehe, i.k. logical OR
&& - loogiline JA tehe, i.k. logical AND
< > == - loogika tehted väiksem, suurem ja võrdne
Nendele vastavate muutujate nimetuste ning funktsionaalsusega saab täpsemalt tutvuda punktis 8.5, kus on juhtprogrammi kood kommenteeritult lahti seletatud.
Lisaks on veel kasutatud alamfunktsiooni measure(), mille ülesandeks on koguda kõikide andurite mõõtetulemused.



Joonis 8.3 Juhtalgoritm

8.5 **Juhtprogrammi koodi koostamine**

Juhtkood on lahti seletatud punktis 8 ning kommenteeritud koodis, seega siinkohal pole palju lisada. Releede koha pealt võib öelda, et nende ümber lülitamine on tehtud madala signaaliga[11]. Ülejäänud osa koodist peaks olema loogiline ning lahti seletatud.

```
/**************************************************
  This is a library example for the MLX90614 Temp Sensor

  Designed specifically to work with the MLX90614 sensors in the
  adafruit shop
  ----> https://www.adafruit.com/products/1748
  ----> https://www.adafruit.com/products/1749

  These sensors use I2C to communicate, 2 pins are required to
  interface
  Adafruit invests time and resources providing this open source code,
  please support Adafruit and open-source hardware by purchasing
  products from Adafruit!

  Written by Limor Fried/Ladyada for Adafruit Industries.
  BSD license, all text above must be included in any redistribution
 **************************************************/

#include <Wire.h>
#include <Adafruit_MLX90614.h>
#include <avr/sleep.h>
#include <avr/interrupt.h>

Adafruit_MLX90614 mlx = Adafruit_MLX90614();

// Konstantide deklareerimine
#define RELAY_ON 0
#define RELAY_OFF 1
int tempPin = 0,gasPin = 1, ventControlPin = 13, wakePin = 2;
```

```
/* Piirtingimused tulekahju signaali sisse lülitamiseks ja ventilaatori
   välja lülitamiseks. Vastavad piirtingimused kalibreeritakse
   testiprogrammi tulemuste alusel  */
double maxgasVal=800, mingasVal = 250, maxtempVal = 60;


// Muutujate deklareerimine

#define CH1  7
#define CH2  8
#define CH3  9
#define CH4  10

// Ventilaatori sisselülitamisel hakkab see pöörlema minimaalse kiirusega.
double controlValue = 5;
double ambTemp, objTemp, roomTemp, gasVal, photoVal;



void setup() {


  // Süsteemi alguses lülitatakse kõik releed välja
  digitalWrite(CH1, RELAY_OFF);
  digitalWrite(CH2, RELAY_OFF);
  digitalWrite(CH3, RELAY_OFF);
  digitalWrite(CH4, RELAY_OFF);

  // Mikrokontrolleri väljundite/sisendite häälestamine
pinMode(CH1, OUTPUT);
pinMode(CH2, OUTPUT);
pinMode(CH3, OUTPUT);
pinMode(CH4, OUTPUT);
pinMode(ventControlPin, OUTPUT);
pinMode(wakePin, INPUT);


// Säästurežiimist väljumiseks vajalik eelndefineeritav käsklus
attachInterrupt(0, wakeUp, HIGH);

  // MLX raamatukogu käivitamine
  mlx.begin();
}
```

```
void loop() {
  // Vastavate sensorite näitude kogumine
   measure();

    //Süsteemi käivitamine
if(objTemp>roomTemp+5){
 // Transistorile algse (minimaalse) PWM suuruse andmine
 analogWrite(ventControlPin,controlValue);

   digitalWrite(CH2,RELAY_ON);  // Käivitab kubu valgustid
       digitalWrite(CH3,RELAY_ON);  // Käivitab ventilaatori
   delay(5000);
}

// Juhtüsteemi töö
while(objTemp>roomTemp+5 && gasVal>mingasVal){

  // Vastavate sensorite näitude kogumine
    measure();

// Osoonigeneraator lülitatakse sisse, kui kubu all hakatakse tööd tegema
  if(gasVal>mingasVal){
  digitalWrite(CH1,RELAY_ON);  // Lülitab osoonigeneraatori sisse
  }
```

```
// Tulekahju kontrollfunktsioon
while(gasVal>maxgasVal || ambTemp>maxtempVal){
 digitalWrite(CH1,RELAY_OFF);  // Lülitab osoonigeneraatori välja
      digitalWrite(CH3,RELAY_OFF);  // Lülitab ventilaatori välja
           digitalWrite(CH4,RELAY_ON);  // Käivitab tulekahju alarmi
gasVal = analogRead(gasPin);
  ambTemp = mlx.readAmbientTempC();
delay(1000);
}
digitalWrite(CH4,RELAY_OFF);  // Lülitab tulekahju alarmi välja

/* 7.5 on samm, mille võrra iga järgnev ventilaatori kiirus eelmisest eristub.
   See on kalibreeritud vastavalt, et aktiivne temperatuuride vahemik 24-58*C
   vastaks PWMi väärtustele 0-255. */

controlValue = (ambTemp-24)*7.5;

// Turvafunktsioon, et mitte anda PWMile suuremat/väiksemat väärtust, kui on lubatud
while(controlValue>255 || controlValue<0){
ambTemp = mlx.readAmbientTempC();
controlValue = (ambTemp-24)*7.5;
delay(100);
}
//Annab transistorile käsu ventilaatori kiiruse reguleerimiseks
analogWrite(ventControlPin,controlValue);

//Osoonigeneraator lülitatakse välja kui kubu all parasjagu tööd ei tehta
if(gasVal<mingasVal){
  digitalWrite(CH1,RELAY_OFF);  // Lülitab osoonigeneraatori välja
  }



}
// Töö lõppedes lülitatakse komponendid välja
 digitalWrite(CH1,RELAY_OFF);  // Lülitab osoonigeneraatori välja
   digitalWrite(CH2,RELAY_OFF);  // Lülitab kubu valgustid välja
       digitalWrite(CH3,RELAY_OFF);  // Lülitab ventilaatori välja
```

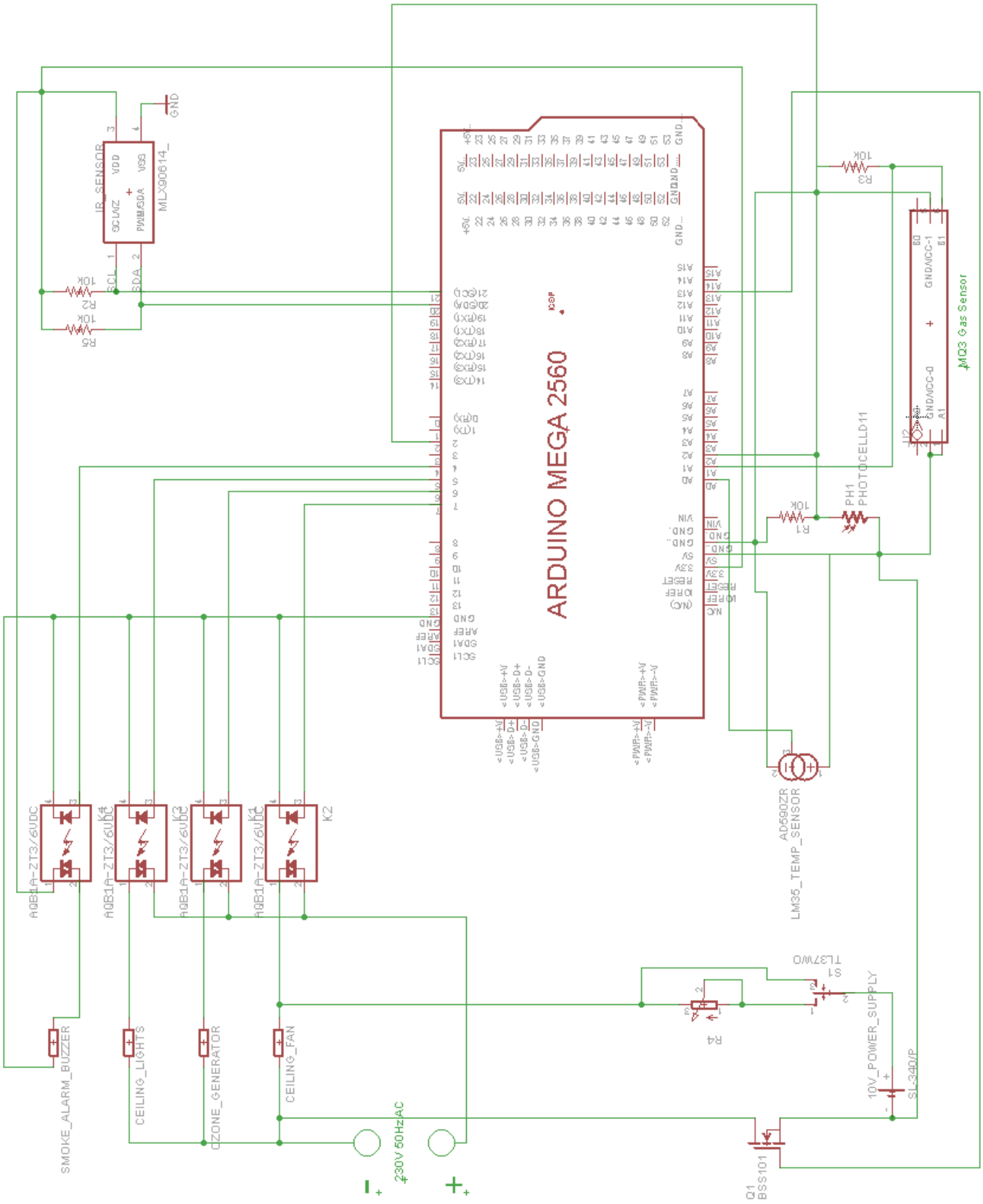
```
// Energiasäästurežiimi lülitamine
if(objTemp<roomTemp+5 && photoVal>800){
sleep();
}

/* Energiasäästmise põhimõttel võetakse andurite lugemeid umbes 2s intervalliga.
   Lisatud vahemik on küll 2s, kuid programmi läbi jooksmiseks kulub kah mingi aeg,
   ent see suurus peaks jääma millisekunditesse  */
  delay(2000);
}

void measure(){
// Vastavate sensorite näitude kogumine
  gasVal = analogRead(gasPin);
  ambTemp = mlx.readAmbientTempC();
  objTemp = mlx.readObjectTempC();
  roomTemp = analogRead(tempPin);
  roomTemp = roomTemp * 0.48828125;
}

// Fototakisti valgustamise korral läheb tööle järgmine funktsioon
void wakeUp()
{
    // Vastav käsklus on vajalik debouncimiseks ehk väärnäitude filtreerimiseks
    detachInterrupt(0);
    /* See funktsioon läheb tööle enne juhtsüsteemi taaskäivitamist.
       Seega on siia võimalik lisada tagasisidet eelnevast töökorrast
       või easisi täiustusi. Hetkel ei ole see aga oluline. */
}
void sleep()
{
    // Siin defineeritakse vastav energiasäästurežiim
    set_sleep_mode(SLEEP_MODE_PWR_DOWN);
    // Vastava raamatukoguga ühendamine
    sleep_enable();
    // Määrab ära katkestamisfunktsiooni
    attachInterrupt(0,wakeUp, HIGH);
    // Siin alles läheb mikrokontroller vastavasse säästurežiimi
    sleep_mode();
    // Katkestuse korral jätkub kood siit
    sleep_disable();
}
```

8.6 **Elektriskeemi koostamine**



Joonis 8.4 Süsteemi elektriskeem

9. Majanduslik osa

Antud töö oli väga mahukas ja aega nõudev ning selle tegemisel esines ka mõningaid anomaaliaid ning sai testitud rohkem seadeid ning andureid, kui lõppversioonis kasutatud oli. Seetõttu on autoril väga keeruline anda adekvaatset hinnangut selle väärtusele.
Tabelis 9.1 on toodud välja kasutatud osade hinnad, mis on ümardatud täisarvudeks, kus võimalik, kuna antud hinnad nagunii kõiguvad ning ei pruugi enam samad olla, kui hankimishetkel ning erinevad tarnijad pakuvad samuti erinevaid hindasid. Antud hindadele lisanduvad veel kaabliühendustele kuluvad summad ning loomulikult paigaldamisel ning programmi tööle panemiseks kuluvad tööajad.
Ventilatsioonisüsteemis olevate komponentide hindu pole välja toodud, kuna see ei puudutanud otseselt autori tööd ning on lihtsalt juhtsüsteemi lisaelementideks. Samuti on hinnaarvutusest välja jäätud kubu enda hind, kuhu süsteem sisse rakendatakse. Kõiki eelnevaid punkte arvestades võiks eeldada, et valmisprojekti hind jääks 300-500 euro vahemikku. Siin kohal võib lisada, et seda süsteemi on loomulikult võimalik odavamalt kokku panna, kui seda ise tegema hakata, kuid vastav isik peab arvestama loomulikult sisse selleks kuluva tööaja alternatiivkuluna.

9.1 Komponentide nimekiri

Kogus	Nimetus elektriskeemil	Kirjeldus	Väärtus	Hind / €
4	R1-R3,R5	Takistid	10kΩ	0.4
1	R4	Potentsiomeeter	10kΩ	0.4
1	Arduino Mega 2560	Arduino mikrokontroller		60
4	K1-K4	1 kanaliga pooljuhtrelee	5V DC	10
1	SMOKE_ALARM_BUZZER	Piesoelektriline summer	3.3-5V DC	1
1	MQ3 Gas Sensr	MQ3 gaasiandur	3.3-5V DC	4
2	MLX90614_	MLX90614ESF-ACF infrapuna temperatuuriandur	3V/5V DC	30
2	LM35_TEMP_SENSOR	LM35 temperatuuriandur	4-30V DC	2
1	PHOTOCELLD11, PH1	Fototakisti	8-20kΩ	0.2
1	Q1	BSS101 MOSFET transistor	240V	5
1	10V_POWER_SUPPLY	Alalisvooluallikas	10V	5
				118

KOKKUVÕTE

Käesoleva bakalaureusetöö teema sai valitud ettevõtte ETS Nordi arendus- ja juhtosakonna koostöös. Nimelt oli vaja välja töötada uus lahendus kommertsköökide ventileerimiseks. Siiani on eesti turul kõik vastavad süsteemid olnud manuaalselt juhitavad ning paratamatult ei ole nende töö optimeeritud vastavalt vajadusele ning keskkonna tingimustele.
Töö eesmärk oli luua automatiseeritud juhtimissüsteem köögikubule, mis võimaldaks reguleerida õhu hulka ning liikumiskiirust vastavalt vajadusele ning kontrollida ventilatsioonisüsteemis olevate komponentide tööd. Kuna autor ise juba teist aastat ettevõttes tööl oli ning talle see projekt ka huvi pakkus, siis võttis ta selle töö enda kätte.
Esmalt sai tutvutud siiani loodud lahendustega ning töökeskkonna parameetritega ning nende alusel kokku pandud skeem, mille välja töötamisega järgnevad viis kuud tegeleda sai. Töö sai tehtud Arduino mikrokontrolleril, millel on paljude muude kontrollerite suhtes eelis, kuna nendele eksisteerib palju informatsiooni ning nende programmeerimine on tehtud kasutajasõbralikumaks. See tegi nende kasutamise hõlpsamaks, kui algselt sai eeldatud.
Analüüsitud sai erinevaid andureid süsteemi valmistamiseks, mille seast valiti välja sobivad temperatuuri- ja gaasiandurid, mis said kõik läbi testitud ja vastavalt süsteemile kalibreeritud.
Järgnevas lõputöö etapis sai loodud mõõtmisprogramm, millega sai kogutud informatsiooni erinevatest töökeskkondadest ning tingimustes, mille alusel tuli välja, et ei ole võimalik teha universaalset süsteemi kõikide köökide jaoks. Seetõttu sai loodud uus mõõtmiskood, mis kogub kokku vastava töökeskkonna parameetrid ning milles alusel saab hiljem juhtimiskoodi modifitseerida.
Juhtimiskoodi tehes ning põhimõtteliselt ka igas eelnevas punktis sai eelkõige rõhku pööratud süsteemi ökonoomilisusele, ehk et seada oleks võimalikult energiasäästlik, ja ergonoomilisusele, et see oleks samal ajal kerge kasutada. Samuti said antud projekti jaoks valitud hästi sobivad komponendid, mis lõpptulemusena väga efektiivse süsteemi kokku tegid.
Niisiis võib öelda küll, et tegemist oli väga huvitava projektiga, mille käigus sai loodud innovaatiline süsteem, mis sobib ideaalselt mistahes suurkööki. Antud töö põhjal saab panna kokku reaalse süsteemi, mida on ka vastavate koodidega võimalik tööle panna. Antud projekt läheb kasutusse ETS Nordi peagi valmiva uue köögikubu koosseisus. Loodetavasti saavad antud projektist kasu mitmed söögikohad.

SUMMARY

Present Bachelor's thesis was chosen in collaboration with ETS Nord's development- and management departments. Namely, it was a necessity to develop a new solution for ventilating commercial kitchens. All the systems that are on Estonian market today are manually controlled and therefore they are not optimized to work correspondingly to environmental conditions and people's needs.

The goal of this work was to create an automated system for the kitchen hood that would be capable of controlling the air flow amount and velocity in the best way possible and thus regulating the work of all the components built into the ventilation system. Because the author had been working in the company for two years and he found the project interesting, he took the task into his own hands.

First of all it was necessary to get familiarized with other solutions created this far and also with the parameters in the working environment on which ground the schematic was created to which the author was engaged with the next five months. The work was created on an Arduino microcontroller, which has many benefits in comparison to many other MCUs, because there is a wide array of information on them and their programming has been made more user-friendly. This made their use easier than it was originally predicted. Different sensors were analyzed in making the system from which the adequate temperature- and gas sensors were chosen. They all got tested and calibrated to work accordingly.

In the next phase the measuring program was created, which was used to gather information in different kind of work environments and conditions. It turned out there was no possible way to make one universal program that works in every kitchen setting. That is why a new measuring program was created, which was objective was to gather the right amount of information, which would be used to modify the control program's work.

As well as in any phase of this project, when writing the final code a lot of emphasis was targeted on the economy and ergonomics. The aim was to make the system as energy saving and easy to use as possible. Furthermore, the components for the project were very well chosen, which made a very effective system in the end.

In conclusion I think that it was a very interesting project during which an innovative system was created, which would suit any commercial sized kitchen. According to these materials one could

create a real working system, which can run with the same codes given in this thesis. This project will be used in a new kitchen hood that is being developed by ETS Nord at the moment. Hopefully this project will be beneficial to many eateries out there.

KIRJANDUS

1. Arduino Mega mikrokontroller: http://www.arduino.cc/en/Main/ArduinoBoardMega2560 (20.05.2015)
2. LM35 temperatuuri anduri infoleht: http://www.ti.com/lit/ds/symlink/lm35.pdf (14.05.2015)
3. Joonise 5.3 allikas ja õppematerjal: http://www.gunnarherrmann.de/blog/temperatur-auslesen-arduino-lm35/ (14.05.2015)
4. MLX90614ESF-ACF temperatuuri anduri infoleht: http://www.adafruit.com/datasheets/MLX90614.pdf (14.05.2015)
5. Arduino raamatukogu ja materjal MLX90614ESF-ACF temperatuuriandurile: https://learn.adafruit.com/using-melexis-mlx90614-non-contact-sensors/wiring-and-test (14.05.2015)
6. Osoonigeneraatori infoleht: http://www.interzon.com/wp-content/uploads/2013/05/Operating-Manual-AirMaid-2000-5000-10000V-110Vac.pdf (24.04.2015)
7. Kubu valgustite infoleht: http://www.osram.com/media/resource/hires/350339/prevaled-coin-pl-cn50-g1.pdf (01.05.2015)
8. Ventilaatori EC Blue infoleht ja dokumendid: http://www.ziehl-abegg.com/ww/fans-download.html?fgr=&dlt=dbl (03.05.2015)
9. Fotoresistori infoleht: https://pi.gate.ac.uk/pages/airpi-files/PD0001.pdf (12.05.2015)
10. Suitsuanduri infoleht: https://www.sparkfun.com/datasheets/Sensors/MQ-3.pdf (13.05.2015)
11. ElectronicsTutorials õppematerjal: http://www.electronics-tutorials.ws/io/io_5.html (13.05.2015)
12. Arduino õppematerjal: http://playground.arduino.cc/Main/LM35HigherResolution (24.05.2015)
13. Arduino õppematerjal: http://playground.arduino.cc/Learning/PhotoResistor (24.05.2015)
14. Arduino õppematerjal: http://playground.arduino.cc/Main/MQGasSensors (24.05.2015)

LISAD

L.1 Mõõtmiskood

Kuna mõõtmiskood tuli väga mahukas, siis sai see toodud välja lisade all.
Vastavalt muutuja sensorType väärtuse kohaselt, salvestatakse vastava anduri väärtused eelnevalt sätestatud mälupesadesse. Kuna seda tehakse kolme erineva anduri tulemustega, oli mõttekam luua alamfunktsioonid, mida oleks võimalik iga anduri lugemiga kasutada. Samuti on koodi lühendamise pärast kutsutud välja teine alamfunktsioon esimese täitmise käigus.

```
/***************************************************
  This is a library example for the MLX90614 Temp Sensor

  Designed specifically to work with the MLX90614 sensors in the
  adafruit shop
  ----> https://www.adafruit.com/products/1748
  ----> https://www.adafruit.com/products/1749

  These sensors use I2C to communicate, 2 pins are required to
  interface
  Adafruit invests time and resources providing this open source code,
  please support Adafruit and open-source hardware by purchasing
  products from Adafruit!

  Written by Limor Fried/Ladyada for Adafruit Industries.
  BSD license, all text above must be included in any redistribution
 ***************************************************/
#include <EEPROM.h>
#include <Wire.h>
#include <Adafruit_MLX90614.h>

Adafruit_MLX90614 mlx = Adafruit_MLX90614();

/*
 Kuna testprogramm korjab andmeid kolme erineva anduri kohtam
millest ühel on kaks erinevat näitu, siis on sensorType muutuja oluline,
et kasutada sama alamprogrammi erinevate sensorite väärtuste salvestamiseks.
Vastavad seosed:
  sensorType = 0; - IR temperatuuri sensori ümbruse temperatuuri mõõtmine
  sensorType = 85; - IR temperatuuri sensori objekti temperatuuri mõõtmine
  sensorType = 170; - LM35 sensori temperatuuri mõõtmine
*/
int sensorType;
// Antud muutuja on kasutusel, et salvestada saadud väärtus õigesse EEPROMi pini.
int storePin = 0;
// Abimuutujad temperatuuri
double ambTemp, objTemp, roomTemp, gasVal, maxgasVal=0;
int tempPin = 0,gasPin = 1;
```

```
void setup() {
// Lülitab sisse IR temperatuuri sensori ning loob sellega I²C ühenduse
mlx.begin();
}

// Mõõtmisprogramm
void loop()
{
// Vastavate sensorite näitude kogumine
  gasVal = analogRead(gasPin);
  ambTemp =mlx.readAmbientTempC();
  objTemp = mlx.readObjectTempC();
  roomTemp = analogRead(tempPin);
//Konstant, millega läbi korrutades saab anduri lugemi Celsiuse skaalal
  roomTemp = roomTemp * 0.48828125;

// Saadud tulemuste salvestamine EEPROM mälusse
  sensorType = 0;
  EEPROMstore(tempRead(sensorType,ambTemp));
  sensorType = 85;
  EEPROMstore(tempRead(sensorType,objTemp));
  sensorType = 170;
  EEPROMstore(tempRead(sensorType,roomTemp));

// Maksimaalse gaasianduri näidu salvestamine
/*
 Kuigi tegelikult on koodis juba EEPROMi 252-255 sisendid kasutusel,
 saame me neid ikkagi gaasianduri tulemite jaoks kasutada, kuna ruumi
 temperatuur ei tohiks üheski keskkonnas tõusta 60 kraadi Celsiuse juurde.
*/
```

```
if(gasVal>maxgasVal){
  double tempVal,tempVal2,tempVal3;
maxgasVal = gasVal;
tempVal = maxgasVal-255;
tempVal2 = tempVal-255;
tempVal3 = tempVal2-255;
if(maxgasVal>255){
EEPROM.write(252,255);
   if(tempVal>255){
  EEPROM.write(253,255);
}
   else{
   EEPROM.write(253,tempVal);
   }
     if(tempVal2>255){
  EEPROM.write(254,255);
}
     else{
   EEPROM.write(254,tempVal2);
   }
     if(tempVal3>255){
  EEPROM.write(255,255);
}
     else{
   EEPROM.write(255,tempVal3);
   }
  }
}

}
```

```
int tempRead(int sensorType, double tempVal){

for(int counter = 0; counter < 1500; counter++){
  // IR ambient temperature test
  if (tempVal < 20){
    storePin = 0+sensorType;
  }
  else if ((tempVal >= 20) && (tempVal < 21)){
    storePin = 1+sensorType;
  }
  else if ((tempVal >= 21) && (tempVal < 22)){
    storePin = 2+sensorType;
  }
    else if ((tempVal >= 22) && (tempVal < 23)){
    storePin = 3+sensorType;
  }
    else if ((tempVal >= 23) && (tempVal < 24)){
    storePin = 4+sensorType;
  }
    else if ((tempVal >= 24) && (tempVal < 25)){
    storePin = 5+sensorType;
  }
    else if ((tempVal >= 25) && (tempVal < 26)){
    storePin = 6+sensorType;
  }
      else if ((tempVal >= 26) && (tempVal < 27)){
    storePin = 7+sensorType;
  }
    else if ((tempVal >= 27) && (tempVal < 28)){
    storePin = 8+sensorType;
  }
    else if ((tempVal >= 28) && (tempVal < 29)){
    storePin = 9+sensorType;
  }
    else if ((tempVal >= 29) && (tempVal < 30)){
    storePin = 10+sensorType;
  }
    else if ((tempVal >= 30) && (tempVal < 31)){
    storePin = 11+sensorType;
  }
    else if ((tempVal >= 31) && (tempVal < 32)){
    storePin = 12+sensorType;
  }
```

```
else if ((tempVal >= 32) && (tempVal < 33)){
  storePin = 13+sensorType;
}
  else if ((tempVal >= 33) && (tempVal < 34)){
  storePin = 14+sensorType;
}
  else if ((tempVal >= 34) && (tempVal < 35)){
  storePin = 15+sensorType;
}
  else if ((tempVal >= 35) && (tempVal < 36)){
  storePin = 16+sensorType;
}
  else if ((tempVal >= 36) && (tempVal < 37)){
  storePin = 17+sensorType;
}
  else if ((tempVal >= 37) && (tempVal < 38)){
  storePin = 18+sensorType;
}
  else if ((tempVal >= 38) && (tempVal < 39)){
  storePin = 19+sensorType;
}
  else if ((tempVal >= 39) && (tempVal < 40)){
  storePin = 20+sensorType;
}
  else if ((tempVal >= 40) && (tempVal < 41)){
  storePin = 21+sensorType;
}
  else if ((tempVal >= 41) && (tempVal < 42)){
  storePin = 22+sensorType;
}
  else if ((tempVal >= 42) && (tempVal < 43)){
  storePin = 23+sensorType;
}
  else if ((tempVal >= 43) && (tempVal < 44)){
  storePin = 24+sensorType;
}
  else if ((tempVal >= 44) && (tempVal < 45)){
  storePin = 25+sensorType;
}
  else if ((tempVal >= 45) && (tempVal < 46)){
  storePin = 26+sensorType;
}
  else if ((tempVal >= 46) && (tempVal < 47)){
  storePin = 27+sensorType;
}
  else if ((tempVal >= 47) && (tempVal < 48)){
  storePin = 28+sensorType;
}
```

```
    else if ((tempVal >= 48) && (tempVal < 49)){
    storePin = 29+sensorType;
  }
    else if ((tempVal >= 49) && (tempVal < 50)){
    storePin = 30+sensorType;
  }
    else if ((tempVal >= 50) && (tempVal < 51)){
    storePin = 31+sensorType;
  }
    else if ((tempVal >= 51) && (tempVal < 52)){
    storePin = 32+sensorType;
  }
  else if ((tempVal >= 52) && (tempVal < 53)){
    storePin = 33+sensorType;
  }
    else if ((tempVal >= 53) && (tempVal < 54)){
    storePin = 34+sensorType;
  }
    else if ((tempVal >= 54) && (tempVal < 55)){
    storePin = 35+sensorType;
  }
    else if ((tempVal >= 55) && (tempVal < 56)){
    storePin = 36+sensorType;
  }
    else if ((tempVal >= 56) && (tempVal < 57)){
    storePin = 37+sensorType;
  }
    else if ((tempVal >= 57) && (tempVal < 58)){
    storePin = 38+sensorType;
  }
    else if ((tempVal >= 58) && (tempVal < 59)){
    storePin = 39+sensorType;
  }
    else if ((tempVal >= 59) && (tempVal < 60)){
    storePin = 40+sensorType;
  }
    else if ((tempVal >= 60) && (tempVal < 61)){
    storePin = 41+sensorType;
  }
  else {
  storePin = 42+sensorType;
  }
}
return storePin;
}
```

```
void EEPROMstore(int sPin){
double pinValue = EEPROM.read(sPin);
int otherPin = sPin+42;
double otherPinValue = EEPROM.read(otherPin);
if(pinValue<255){
pinValue=pinValue+1;
}
else
{
 otherPinValue = otherPinValue + 1;
 pinValue = 0;
 EEPROM.write(otherPin, otherPinValue);
 }

EEPROM.write(sPin,pinValue);
}
```

L.2 Mõõtetulemuste lugemine EEPROM mälust

Tegemist on lihtsalt lühikese koodiga, mis kuvab Arduino konsoolis eelneva mõõtekoodi poolt saadud tulemused.

```
#include <EEPROM.h>

int address = 0, sensorType;
int i = 20, j = 21;
double maxgasVal;

void setup()
{
  // Kasutajaliidese sisselülitamine
  Serial.begin(9600);

}

void loop()
{
  // Temperatuuri andurite tulemuste lugemine

  if(address==85)
  sensorType=85;
  if(address==85)
  sensorType=127;
  if(address==170)
  sensorType=209;
```

```
  if(sensorType == 42){
  Serial.println("IR ambient readings");
  Serial.print("<20*C : ");
  }
  if(sensorType == 127){
  Serial.println("IR object readings");
  Serial.print("<20*C : ");
  }
  if(sensorType == 209){
  Serial.println("LM35 readings");
  Serial.print("<20*C : ");
  }
  Serial.println(EEPROM.read(address));
  while(address<43+sensorType){
  Serial.print(i,DEC);
  Serial.print(" - ");
  Serial.print(j,DEC);
  Serial.print("*C : ");
  Serial.println(EEPROM.read(address));
  address=+1;
  i=i+1;
  j=j+1;
  if(i==61 && j==62){
  Serial.print(">61*C : ");
  Serial.println(EEPROM.read(address));
  i=20;
  j=21;
    }
    }
  // Gaasianduri tulemuse lugemine
  if(sensorType == 209){
    maxgasVal= EEPROM.read(252) + EEPROM.read(253) + EEPROM.read(254) + EEPROM.read(255);
    Serial.println("Max gas sensor value: ");
  Serial.print(maxgasVal,DEC);
  }
}
```