Alex Malinen

Näyttökalvon puhdistuslaitteen kehitys

Metropolia Ammattikorkeakoulu

Ylempi AMK -tutkinto

Älykäs teollisuus

Opinnäytetyö

28.4.2025

Tiivistelmä

Tekijä(t):	Alex Malinen
Otsikko:	Näyttökalvon puhdistuslaitteen kehitys
Sivumäärä:	35 sivua + 2 liitettä
Aika:	28.4.2025
Tutkinto:	Ylempi AMK -tutkinto
Tutkinto-ohjelma:	Älykäs teollisuus
Suuntautumisvaihtoehto:	-
Ohjaaja(t):	Lehtori Matti Välikylä

Lopputyössä kehitettiin näyttöpäätteelle soveltuvaa kalvonpuhdistuslaitetta toimivaksi prototyypiksi aina simuloinnista kokoonpanoon ja testaamiseen. Laiteen ohjaus toteutettiin releisiin kytketyllä mikro-ohjaimella, joiden tilaa vaihdettiin mikro-ohjaimeen kytketyn infrapunaa käyttävän liiketunnistimen tietojen perusteella. Releiden kautta pyöritettiin näyttöpäätteen päällä olevaan kalvoa sähkömoottorin avulla sekä johdettiin virta bakteereja tuhoavalle valolle kalvon puhdistusta varten. Mikro-ohjaimen ohjelma aloittaa puhdistuksen, kun näyttöpäätteellä ei havaita käyttäjää määritetyn ajan kuluessa ja ohjelma myös osaa lopettaa puhdistuksen, jos kesken kaiken käyttäjä tulee laitteelle. Mikro-ohjaimen ohjelma on kirjoitettu ja kommentoitu siten, että liikeanturin herkkyyttä ja muiden toiminnallisuuksien ajoituksia voidaan säätää.

Avainsanat: PIR-anturi, Arduino, Sähkömoottori, Hygienia, Kosketusnäyttö, Tinkercad

---

Abstract

Author(s):	Alex Malinen
Title:	Development of cleaning a screen film
Number of Pages:	35 pages + 2 appendices
Date:	28 April 2025
Degree:	Professional Master´s degree
Degree Programme:	Intelligent Industrial Solutions
Specialisation option:	-
Instructor(s):	Matti Välikylä, Senior Lecturer

In the thesis a film cleaning device suitable for touch screen was developed to working prototype all the way from simulation to assembly and to testing. Device was controlled by relays which were connected to micro controller and which state was switched according to infrared based motion detection sensor signals. Through relays the film was rotated around the touch screen by using DC electric motor and through the relays current was also directed to the light for cleaning the film. Microcontroller starts the cleaning when motion sensor does not detect the user after set time and program also knows to stop cleaning if user appears to the machine. Microcontroller program is written and commented in a way that motion sensor sensitivity and other functional timing can be adjusted.

Keywords: PIR-sensor, Arduino, Electric motor, Hygiene, Touch screen, Tinkercad

Sisällys

1 Johdanto

Lopputyön aiheena on kehittää prototyyppilaitetta ja saada aikaan kaupalliseen loppusovellukseen sopiva piirikortti, joka täyttää työntarjoajan vaatimukset toiminnallisuuksiensa puolesta. Kortti ohjaa eri antureilta saatujen signaalien perusteella ulostuloja, jotka kytkevät päälle eri toimintoja. Työn tilasi ClearReino Oy, joka on vuonna 2022 perustettu start-up-yritys. ClearReino Oy keskittyy kosketusnäyttöjen hygienian ylläpitämiseen.

Laite, johon piirikorttia suunnitellaan, on kosketusnäyttöpääte. Näytön ympäri on vedetty kalvo, joka ajetaan sähkömoottorilla näytön taakse puhdistukseen käyttökertojen välissä. Tarkoituksena on estää kosketusnäytön pintaan jääviä epäpuhtauksia leviämästä seuraavaan käyttäjään. Näytön lähelle on asennettu liiketunnistin, jonka perusteella kortti ohjaa puhdistustoimintoja.

Kortti lukee liiketunnistimen signaalia jatkuvasti ja tunnistaa sen perusteella, milloin näyttöpääte on käytössä. Kun viimeisestä havaitusta tunnistuskerrasta on kulunut tietty aika, ajetaan kalvo laitteen taakse ja kytketään päälle valo, jolla puhdistus suoritetaan. Moottoria tai valoa ei kuitenkaan saa ajaa silloin kun tulkitaan, että päätteelle tulee henkilö vaan toiminta pysäytetään turvallisuus ja käyttäjäystävällisyyden vuoksi, kunnes henkilö poistuu paikalta. Jos päätteellä ei tarpeeksi pitkään aikaa tunnisteta ketään, tehdään puhdistus säännöllisin väliajoin.

Lopputyötä varten tehtiin tutkimussuunnitelma, johon arvioitiin työvaiheet ja tavoitteet pääpiirteittäin:

1. Vuoden 2022 loppuun mennessä kehittyneempi versio prototyypistä valmiina niiltä osin, joihin osallistun. Laitteen ohjainkortti on joko pienempi sisältäen samat ominaisuudet kuin proto tai toiminnallisuuksia lisätään koon pysyessä tilaajayrityksen määrittämissä mitoissa tai molempia.

2. Laite on testattu toimivaksi ja luotettavaksi ja sisältää ne toiminnallisuudet, joita tilaajayritys määrittää.

3. Laitteen ohjaimelle on tuotettu käyttöohjeet, joilla jatkokehitystä pystytään tekemään helpommin ja vikatilanteita ymmärretään paremmin.

Tehtävä	**Arvio valmistumiselle**
Prototyypin kasaaminen ja toimintaan tutustuminen	7/2022
Olemassa olevien toiminnallisuuksien määrittäminen ja kehityskohteiden listaaminen	7/2022
Elektroniikkasuunnitteluun ja/tai ohjelmointikirjallisuuteen syventyminen	8/2022
Prototyyppilaitteen komponenttien optimointimahdollisuuksien määrittäminen	8/2022
Mikro-ohjaimen tai muun ohjainlogiikan ohjelmointi ja valittujen komponenttien simulointi	9/2022
Piirikorttisuunnittelu valittujen komponenttien pohjalta, komponenttien hankinta. Tiedostojen tuottaminen	9/2022
Piirikortin valmistaminen ja komponenttien kiinnitys sekä muut liitännät	10/2022
Valmistetun tuotteen testaaminen ja toiminnan varmistaminen, Laitteen käytön ohjeistus	11/2022

Taulu 1. Työn aikataulusuunnitelma.

1.1 ClearReino Oy

Vuonna 2018 julkaistiin Teknologian tutkimuskeskus VTT Oy:n tutkimus lentokentillä esiintyvistä bakteereista osana Euroopan unionin rahoittamaa PAND-HUB-projektia (Preventation of High Threat Pathogen Incidents in Transport Hubs). Tarkoituksena oli ymmärtää miten bakteerien ja niiden aiheuttamat taudit leviävät, jotta vakavassa epidemiassa osataan kohdistaa ihmisten suojeluun tarkoitettuja toimenpiteitä tehokkaitten. Tutkimuksessa todettiin, että käytännössä jokaiselta kosketuspinnalta löytyy laaja kanta erilaisia bakteereja (Liite 1). Reino Malinen sai tutkimustuloksesta idean hygieenisestä kosketusnäytöstä,

joka pohjautuisi ultraviolettivalolla puhdistamiseen. Huomattuaan ettei vastaavalle keksinnölle ollut patenttia, idea jalostettiin suunnitelmaksi laitteesta. Mukaan pyydettiin Peter Christiansen, jolla oli kokemusta teollisten hygieniaratkaisujen kaupallistamisesta sekä Pekka Päivärinta, jolla puolestaan oli kokemusta ohjelmisto- ja tuotekehitysyrityksen perustamisesta sekä toimitusjohtajana toimimisesta.

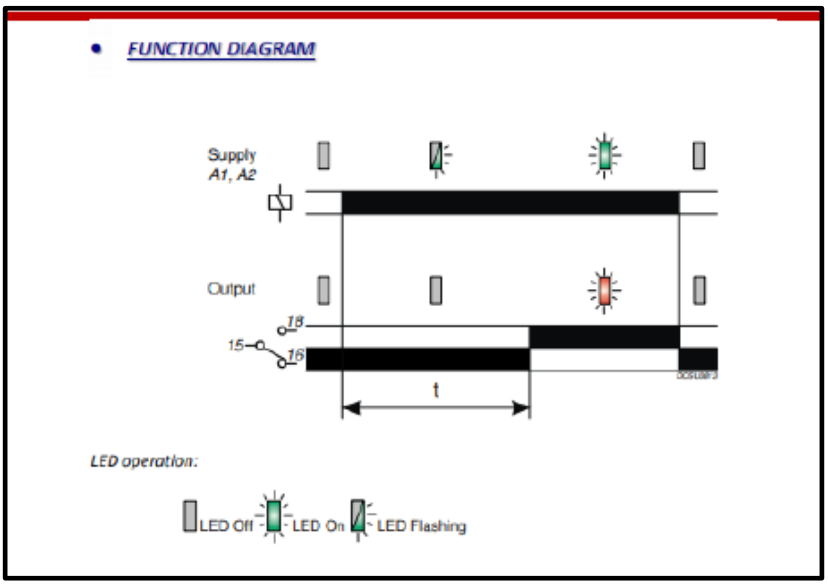
Reino, Peter ja Pekka perustivat ClearReino Oy:n vuonna 2022 ja aloittivat keksinnön patentoinnin, joka myönnettiin 2023 Suomessa ja myöhemmin samana vuonna EU:ssa. 2023 keväällä valmistui ensimmäinen prototyyppi ja yhteistyö VTT:n kanssa aloitettiin, jotta voitaisiin varmistua laitteen toimivuudesta. VTT:n mittauksissa laite yllätti tutkijat tehokkuudellaan tuhoten bakteereja paremmin kuin uskottiin. Keksinnölle on löytynyt mielenkiintoa ja sitä on esitelty mahdollisille asiakkaille useaan otteeseen. Esittelytilaisuuksista saadaan arvokasta palautetta laitteen jakokehitystä varten. (Liite 2)

2 Prototyyppi

2.1 Lähtötilanne

Työn alussa osa laitteen toiminnoista oli jo kokeiluasteella. Laitteen oli ajateltu toimivan pääasiassa ajastinreleillä, ja että mikro-ohjain huolehtisi vain liikeanturin signaalin lukemisesta ja antaisi sen perusteella luvan ajastinreleille toimia. Ajastinrele toimii kuten normaalirelekin, mutta releelle on mekaanisesti säädettävissä esimerkiksi tietty aika mitä rele odottaa ennen kytkeytymistään päälle.

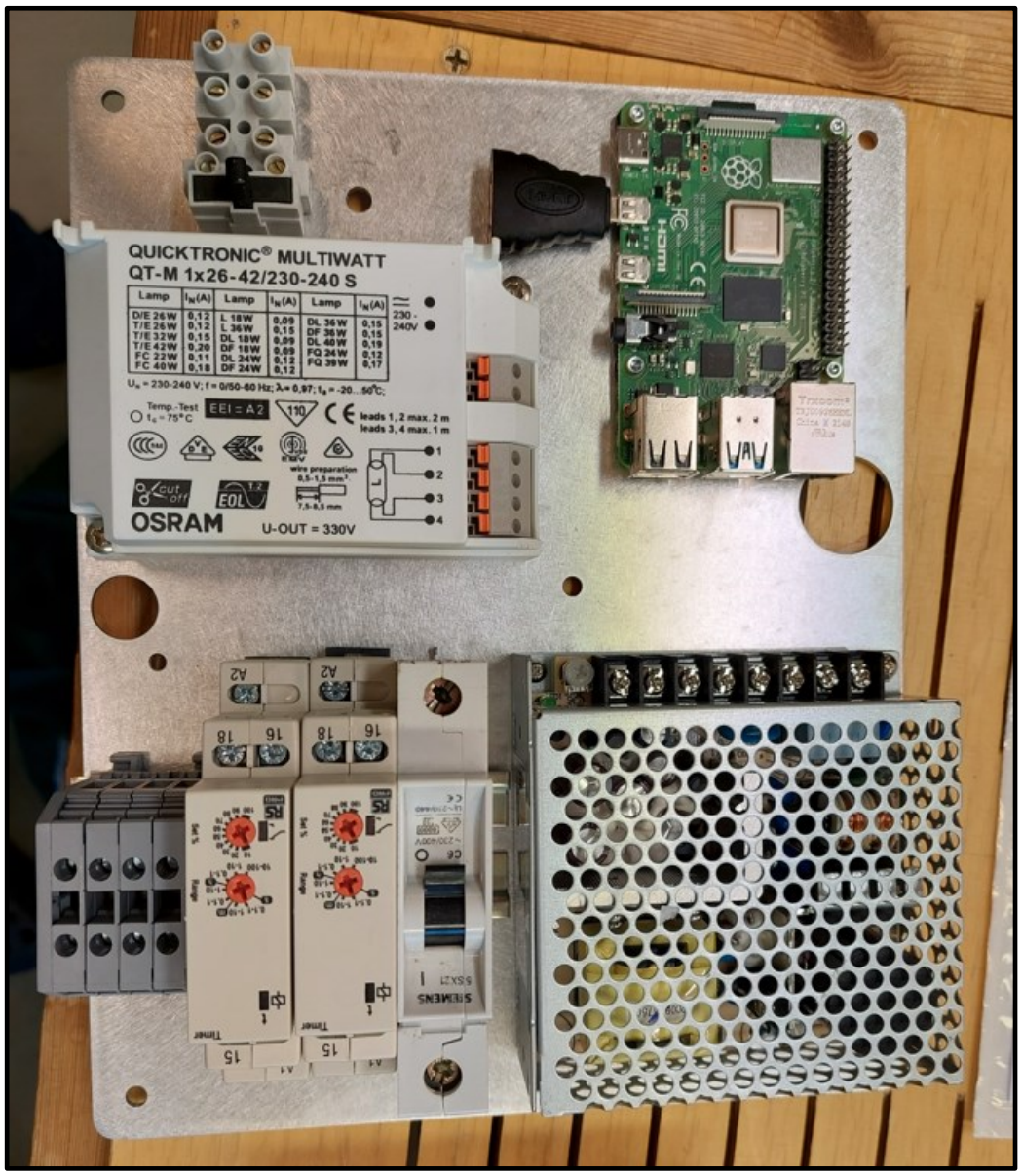
Ajastinreleitä oli aluksi kytketty sarjaan, jolla pyrittiin hoitamaan kalvomoottorin ja puhdistusvalon toimintaa halutulla tavalla.



Kuva 1. Toimintadiagrammi ajastinreleelle RS PRO SPDT [1].

Releiden kautta ohjattiin 12 voltin tasavirta sähkömoottoria, jonka tehtävä oli pyörittää kalvoa näytön ympäri. Tarkoitus oli ajastireleillä säätää, kauanko moottoria ajetaan, jotta saadaan koko kosketuspinta siirrettyä näytön taakse

puhdistukseen. Ajastinreleillä oli myös tarkoitus ohjata, kauanko valoa pidetään päällä puhdistusta varten.



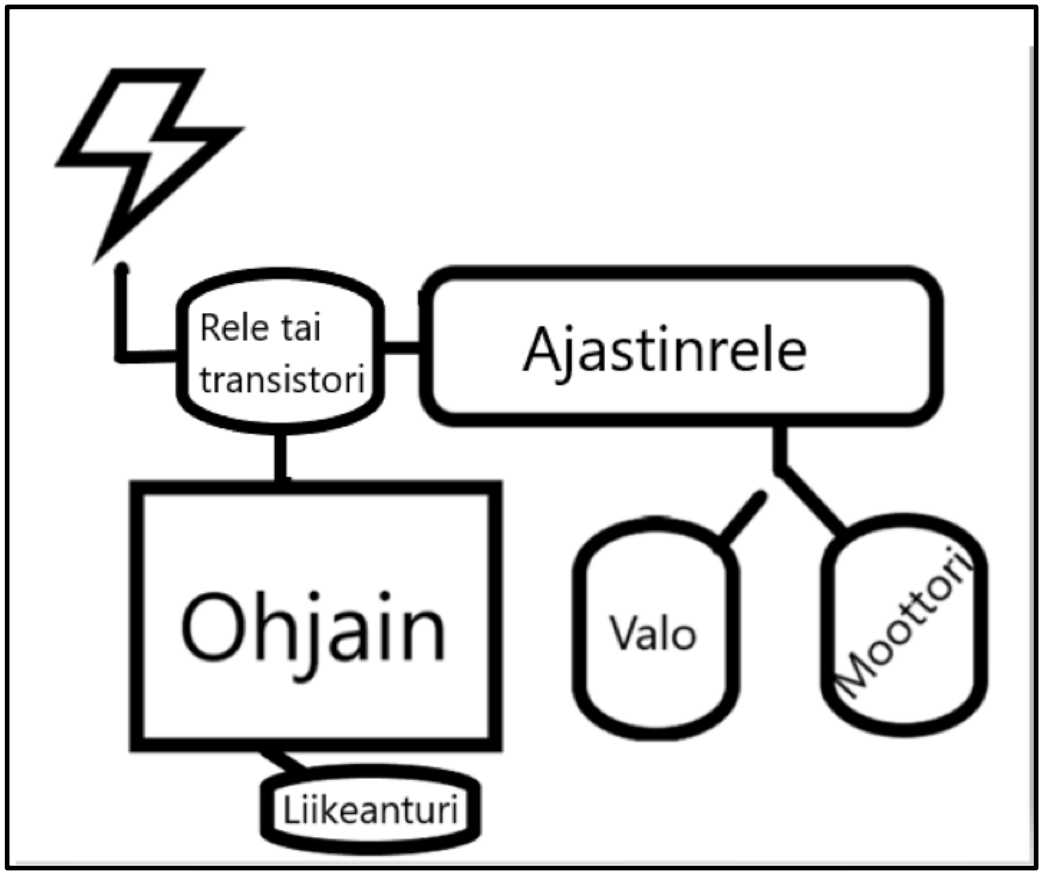
Kuva 2. Komponenttien sijoittelua alkuvaiheessa

Ajastinreleen toiminta oli kuitenkin ymmärretty hieman väärin eikä laitetta oltu päästy pitkälle testaamaan. Ajatuksena oli, että liikeanturi ohjaisi ajastinrelettä siten, että kun liiketunnistin ei tunnista niin rele lähtisi laskemaan säädettyä aikaa, jonka jälkeen kalvomoottoria ajettaisiin. Tämä ei toimi sillä signaalitulo liike-

anturissa on avoin kollektori eli toimii käänteisesti, miten oli alunalkajaan ajateltu eikä suoraan syöttäisi käyttöjännitettä releelle. Vaikka signaalin anturilta saisikin suoraan releelle niin sehän muuttuisi sitä mukaan, kun anturi tunnistaisi liikettä. Mikro-ohjaimen kauttakin releiden ohjaus vaatii muitakin komponentteja sillä mikro-ohjaimelta ei pääsääntöisesti saa suoraan ohjattua suurempia releitä jännite ja virtarajoitusten vuoksi. Epäselvää oli myös, jos rele saadaan kytkettyä päälle oikeaan aikaan tästä huolimatta ja moottoria ajetaan tietyn odotusajan jälkeen, niin palautuuko releen asetus lepotilaan tietyn ajan kuluttua. Ajastinreleiden ohjelehtisistä selvisi, että ei palaudu, sillä releessä ei voida hallita mitään muuta kuin odotusaikaa siitä hetkestä, kun lähteelle tulee virta. Rele odotusajan jälkeen, muutettua ulostuloaan, jää siihen muutettuun asentoon, kunnes lähteen virta katkaistaan ja kytketään uudestaan. Tämän lisäksi haluttiin samalla toisella ajastinreleellä pilkkoa vielä ensimmäisen toiminta tunnin välein tehdyksi puhdistukseksi, vaikka ihmisiä ei olisikaan käynyt laitteen lähettyvillä. Ehdotin kuitenkin jo aikaisessa vaiheessa, että ajastustoiminnot yritettäisiin hoitaa mikro-ohjaimen kautta, sillä vastaavasta oli hieman kokemusta aiemmista töistä. En nähnyt mielekkääksi jakaa ohjausta koodin ja mekaniikan välille sikäli, kun vain toinen riittäisi, sillä nähdäkseni tämä tekisi toteutuksesta tarpeettoman monimutkaista. Keskusteltuamme asiasta sain luvan lähteä toteuttamaan ajastusta koodin puolella. Ajatus ja etu ajastinreleiden kanssa oli siinä, miten protoiluvaiheessa olisi mekaniikkapuolella työskentelevien mahdollisuus säätää ajastinreleen ruuvista yksinkertaisemmin moottorin ja valon ajoaikaa tarpeen mukaan, sillä oli luonnollista olettaa, että näitä tarvitsee säätää useasti, jotta sopivat asetukset löytyvät. Emmehän tietäneet esimerkiksi kuinka raskas kuorma moottorilla olisi siirrettävänään. Kuorma sitten vaikuttaa mahdollisesti moottorin nopeuteen, joka sitten pitäisi ehtiä pyöräyttämään käytettyä osa kalvosta kokonaisuudessaan puhdistuspuolella ja samalla tietysti puhdistettu pinta valmiina käytettäväksi. Sama juttu valon kanssa eli vaati tutkimusta, jotta selvisi, kuinka pitkään valoa tulee pitää päällä, jotta puhdistus on tehokasta ja bakteerit kuolevat.

Alussa oli myös toiveita, että mikro-ohjain osaisi tallentaa virhetilanteita tai muuten valvoa järjestelmän toimivuutta. Toivottiin myös, että lokia kirjattaisiin esimerkiksi pilvipalveluun, josta sen tietoja voisi lukea toimistolta riippumatta missä

laite on asennettuna. Tällaista en ollut ennen tehnyt, muuta kuin yksittäisellä koulun kurssilla, mutta uskon, että olisi ollut tehtävissä. Kuitenkin esitettyäni muutamia tarkentavia kysymyksiä, kuten: ”yhdistetäänkö laite sitten asiakkaan verkkoon?” ja ”mitä tietoja halutaan kirjata tai mikä tilanne tulkitaan virheeksi” asiaan ei juurikaan palattu.

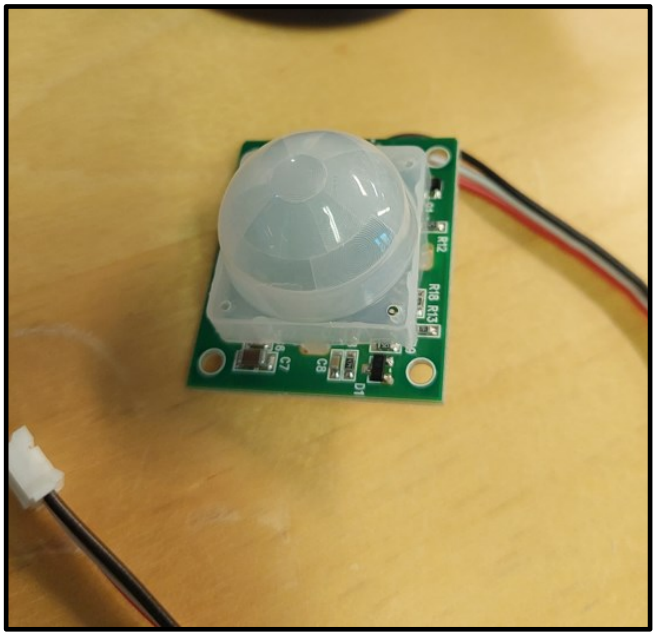


Kuva 3. Aikaisen kehitysversion luonnos

2.2 Liikeanturi

Liikeanturina käytimme PIR-anturia eli passiivista infrapuna-anturia. Anturin toimintaperiaatteena on, että kaksi anturin osaa vastaanottaa infrapunasäteilyä tunnistuselementtiin infrapunaa läpäisevän ikkunan kautta. Elementti on hermeettisesti suljetussa kotelossa, jotta sen näkemä säteily ei häiriintyisi äänistä, lämpötilasta tai kosteudesta. Anturin kahden osan vastaanottama säteilyn ollessa sama, signaali pysyy vakiona ja voidaan tulkita, ettei luenta-alueella ole

liikettä. Jos taas lämmin keho, joko ihmisen tai eläimen, liikkuu tunnistusosien välistä, niin anturi havaitsee saadun säteilyn eron osissa, joka taas aiheuttaa muutoksen ulosluettavassa signaalissa ja tulkitaan liikkeeksi [2].



Kuva 4. PIR-Liikeanturi SE-10 [3].

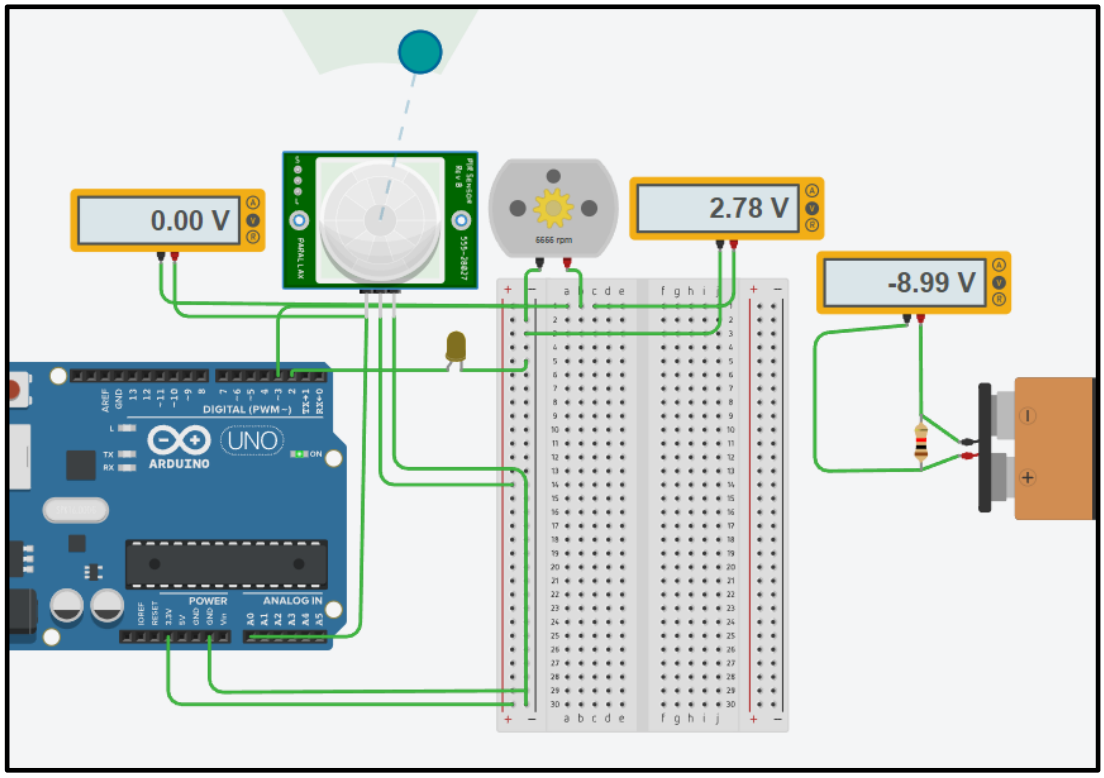
Proto 1 liikeanturiksi valittiin edullisemmasta päästä oleva SE-10 joka toimii 12V käyttöjännitteellä eli vaati johdotuksen ulkoiselta virtalähteeltä. Anturin hinta on noin 10 euroa, joten näitä oli hankittu useampia valmiiksi siltä varalta, että testien aikana sattuisi haavereita ja antureita menisi rikki. Ilmeisesti näin oli käynyt-kin ennen mikro-ohjaimen lisäämistä prototyyppiin ja ainakin 3 anturia oli todettu toimimattomiksi tai epäluotettaviksi, mutta jäi vähän epäselväksi millä tavoin vahinko oli saatu aikaiseksi. Omissa testeissäni en saanut ainuttakaan rikki, mutta havaitsin osia tarkastellessani, että anturin laadunvalvonnassa voisi olla parannettavaa. Toisessa saadussa anturissani oli kiitettävän pitkä metallilanka piirikortin suojakalvon alla, joka visuaalisesti ei onneksi tehnyt oikosulkua johdinpin-

tojen välillä eikä päässyt liikkumaan mihinkään. Testasin tätäkin anturia myöhemmin erikseen mielenkiinnosta ja toimi nähdäkseni normaalisti, mutta ei herättänyt varsinaisesti luottamusta valmistajaa kohtaan.

2.3 Simulointi

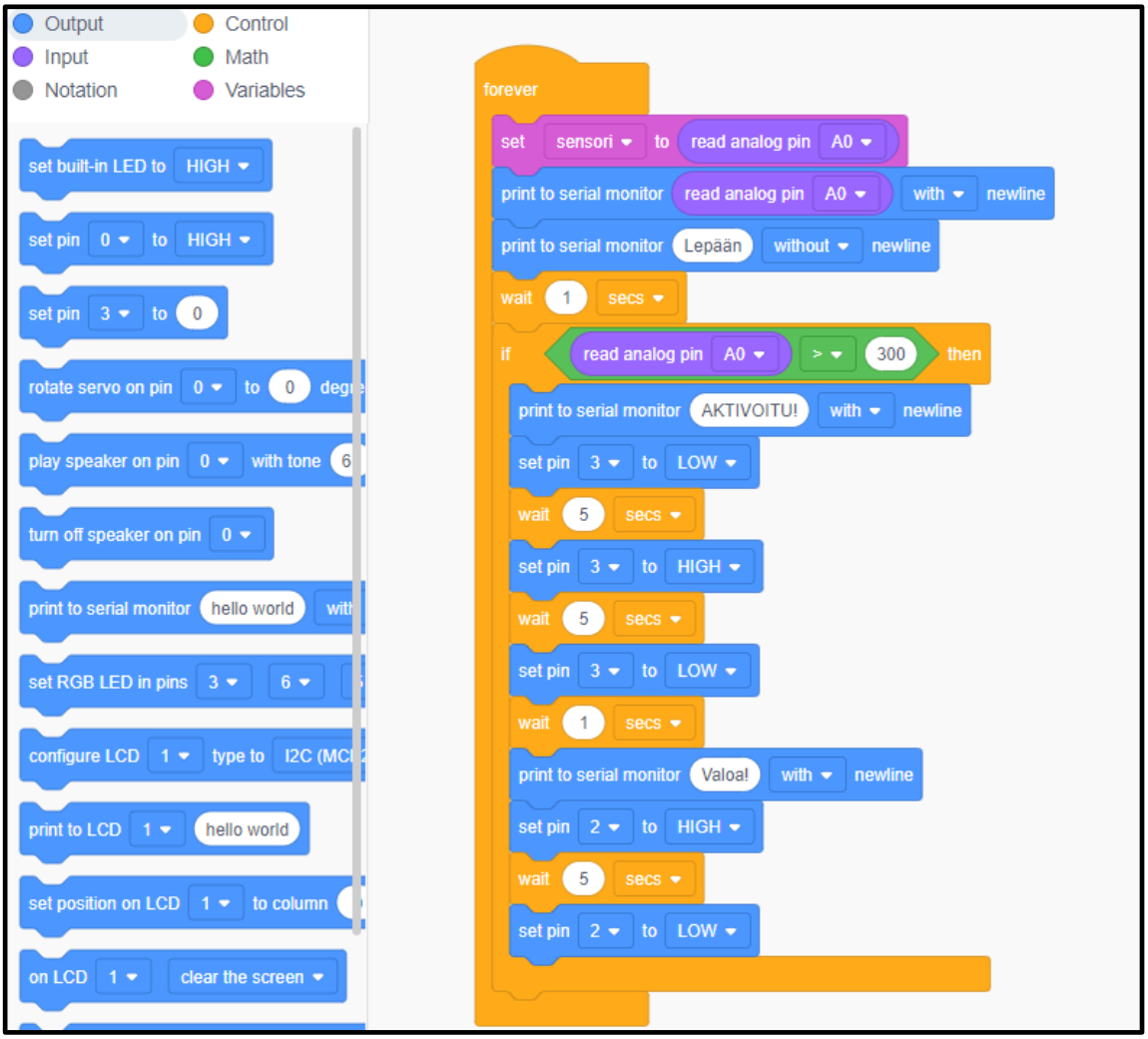
Aloitin suunnittelun etsimällä verkosta sopivaa simulointiohjelmaa, jossa voisin hahmotella miten laite voisi toimia. Internetistä löytyy lukuisia ilmaisia ohjelmia, jotka on tarkoitettu hyvinkin pitkälti vastaavaan protoiluun kuin mitä tässä työssä lähdettiin tekemään. Varmaankin mikä tahansa olisi sopinut omaan käyttöön, mutta päädyin Autodeskin Tinkercad – ohjelmaan, joka mainosti itseään helppona ja suosittuna sekä lupaili mahdollisuuden piirisuunnitteluun ja ohjelmointiin.

Tinkercadissa valitaan komponentit valmiista kirjastosta. Kirjastosta löytyy nähdäkseni kaikkia peruskomponentteja muutamaa erilaista eikä tässä vaiheessa tarkkoja malleja ollutkaan valittu. Esimerkiksi laitteen puhdistusvaloa simuloin vain led-valolla, joka riitti minulle hyvin näyttämään, että simulaation koodi toimii kuten ajateltu. Aivan täydellinen ohjelma ei ollut, sillä osa kirjaston komponenteista eivät jostain syystä toimineet laisinkaan, vaan heittivät simulaatiota käynnistäessä virheviestin, ettei piiriä voi simuloida. Uskoin aluksi, että tämä olisi jokin johdotusvirheeni ohjelmassa, mutta usean yrityksenkään jälkeen en saanut ongelmaa ratkeamaan. Tinkercad ei myöskään osannut kertoa mikä valitsemistani komponenteista virheen aiheutti, joka johti tuskastuttavaan simuloinnin kokeiluun aina poistamalla yksi osa piiristä kerrallaan. Onneksi osia oli verrattain vähän ja kuten edellä mainittu kirjastossa oli useampia vastaavia komponentteja valittavana, joista jokin aina loppujen lopuksi toimi.



Kuva 5: Komponentteja Tinkercad-ohjelmassa [4].

Tinkercadissa lisätylle mikro-ohjaimelle voitiin rakentaa koodi myös valmiin kirjaston ohjelmointipalikoista. Tämä tuntui aloittelevalle koodarille todella selvältä tavalta nopeasti kokeilla millainen ohjelman logiikka voisi olla. Palikkojen asettelu ja toiminta tuntui hyvin selkeältä ja omatoiminen harjoittelu riitti ymmärtämään kutakin toimintoa. Varsinkin jos kirjoittaisin täysin tuntemattomalla koodikielellä olisi simuloinnin koodipuolessa voinut mennä tuntikaupalla aikaa sillä, koodin kieliasu on yleensä hyvinkin pilkuntarkka ja vaatii keskittymistä sekä opiskelua ennen kuin yksinkertaisempikaan toiminto saadaan kirjoitettua. Ohjelmointipalikoita oli kirjastossa hyvin omiin käyttötarkoituksiini, mutta voin kuvitella, että monimutkaisempi ohjelma ei välttämättä suoraan näillä työkaluilla onnistuisi.



Kuva 6: Ohjelmointi Tinkercadissa

Tinkercad auttoi myös varsinaisen mikro-ohjaimen koodin kirjoituksessa alkuun, sillä ohjelmointipalikoista tehty koodi voitiin ladata talteen. Koodi muuttui tässä yhteydessä tekstimuotoon, joka auttoi ymmärtämään millainen ensimmäisistä varsinaisista versioista voisi tulla. Ladattu koodi tuli C++ kielellä kirjoitettuna eli sitä ei suoraan voitu kopioida valitulle mikro-ohjaimelle, mutta onneksi käyttämieni ohjelmointikielien rakenne ei eroa merkittävästi esimerkistä.

```
// C++ code
//
int sensori = 0;

void setup()
{
  pinMode(A0, INPUT);
  Serial.begin(9600);
  pinMode(3, OUTPUT);
  pinMode(2, OUTPUT);
}

void loop()
{
  sensori = analogRead(A0);
  Serial.println(analogRead(A0));
  Serial.print("Lepään");
  delay(1000); // Wait for 1000 millisecond(s)
  if (analogRead(A0) > 300) {
    Serial.println("AKTIVOITU!");
    digitalWrite(3, LOW);
    delay(5000); // Wait for 5000 millisecond(s)
    digitalWrite(3, HIGH);
    delay(5000); // Wait for 5000 millisecond(s)
```

Kuva 7: Koodin lataaminen Tinkercadista

2.4 Prototyypin kehitys

Ensimmäisen version kokoonpanovaiheeseen ostettiin määrittelemiäni komponentteja elektroniikkaa varten. Kasa vastuksia suojelemaan, ettei Arduino varmasti vaurioituisi korkeammista virtamääristä ja epäilin, että joutuisin taas kelluvien signaalien käsittelyn opettelemaan uusiksi myös. Hankimme myös muutaman leipälaudan eli koekytkinlevyn, jonka avulla on helpompi kokeilla ja ennen kaikkea purkaa johtoliitoksia ilman juottamista. Lisäksi hankimme useita releitä ohjaamaan moottoreita ja muuta mahdollista.

2.4.1 Releet

Valitsin releiksi SSR-releitä eli kontaktittomia releitä. Valitsin releet sillä perusteella, että niiden kytkentään riittäisi Arduinon 5 volttia suoraan eikä olisi tarvetta erikseen transistoreilla kytkeä releitä päälle ja pois ja ottaa releiden käyttöjännite jostain muualta. Periaatteessa tähän olisi soveltunut varmasti muunkinlaiset

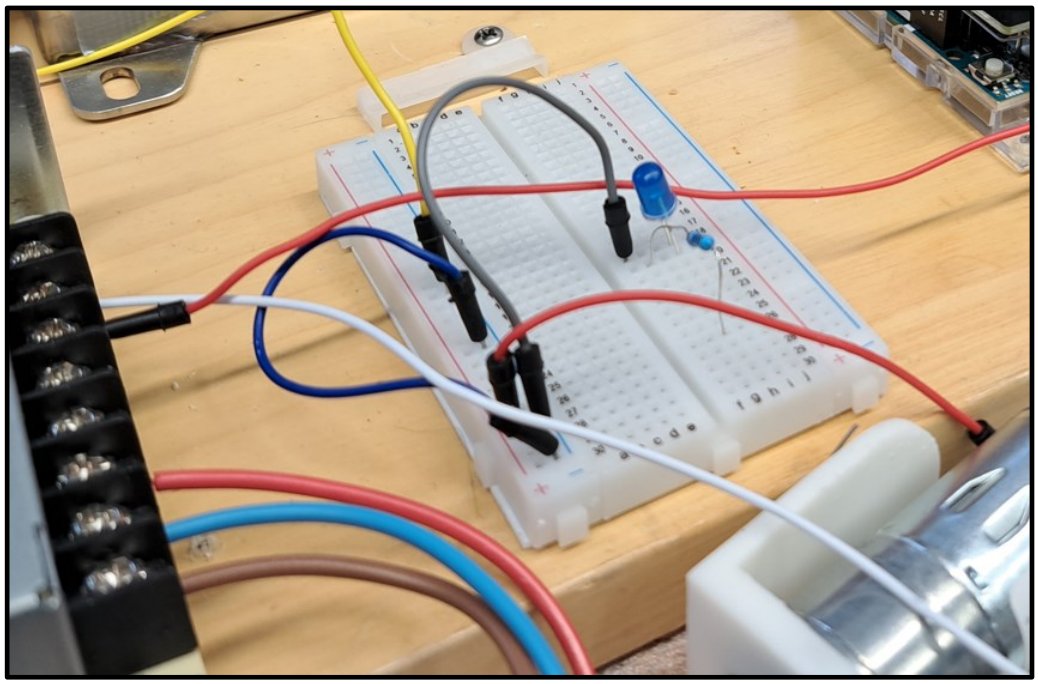
releet, mutta luin että lueskelin SSR-releiden olevan varsin pitkäikäisiä ja hiljaisia johtuen siitä, ettei rele sisällä liikkuvia osia kuten perinteiset releet. Releen kytkin toimii siis täysin elektronisesti. Varsinkin suuremmiksi mitoitetuilla releillä kytkentä-ääni voi olla kova ja tahdoin ettei suurta kytkentä-ääntä haluttaisi mahdollisissa prototyypin esittelytilaisuuksissa saatikka loppusovelluksessakaan sillä elektroniikka on suhteellisen lähellä käyttäjää ja voisi sitten heikentää käyttökokemusta. Tilasimme varmuudeksi molempia tasavirta ja vaihtovirtaan soveltuvia releitä, jos esimerkiksi vaihtovirran käytölle tulisi tarpeita lampun ohjaamisen kanssa.

Releiden kytkemisessä Arduinoon olin vähän varovainen sillä mittasin vastuksen olevan vain vähän reilun Ohmin joka 5V käyttöjännitteellä menisi sitten reippaasti Arduinon kestämän virtamäärän yli, joten rajoitin tätä ensi 1000 Ohmin vastuksella. Tämäpä ei suoraan toiminut koska rajoitti virtaa liikaa eikä rele sitten jaksanut muuttaa kytkimensä asentoa. Releen datalehdestä tarkistettuna jäin juuri hieman vaaditun minimin 5 mA alle ja vaihtamalla vastuksen pienempään 500 Ohmiin rele lähti toimimaan halutulla tavalla.

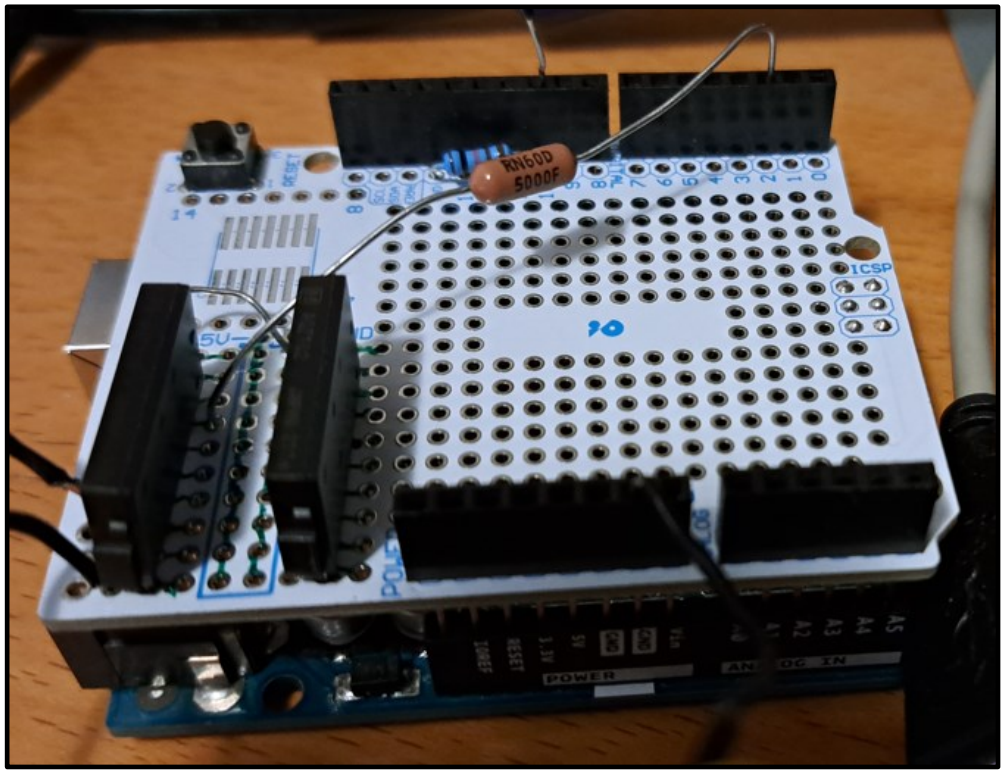
2.4.2 Koekytkinlaudat

Koekytkentälautoja hankittiin muutamia erimallisia. Ensimmäinen lauta oli SAD-01 joka on vain johdinpintojen päälle asetettu muovinpala, johon saa helposti tökättyä johtimia ja sopivan kannan omaavia peruskomponentteja päälle. Datalehtisestä ja laudasta itsestään näkee selvästi mitkä reiät ovat kytkettyinä mihinkin ja kokeilu ja vianetsintä, kun menee pieleen, on helppoa. Liitettävän koekytkentälevyn hyöty vähenee heti kun valmista elektroniikka aletaan asentamaan laitteeseen sillä se vie runsaasti tilaa, ja johdot sekä komponentit lähtevät herkästi irti levystä. Tilasimme myös toisen koekytkentälevyn, joka oli suunniteltu Arduinon päälle suoraan asetettavaksi. Arduinon koekytkentälevy erosi ensimmäisestä myös siten, että siihen komponentit tuli juottaa kiinni eli tämä oli hyvä ratkaisu ensimmäisen levyn ongelmiin. Säästimme tilaa ja komponentit eivät varisseet pois jokaiset tärähdyksen myötä. Ongelma tässä levyssä oli, ettei sille löy-

tynyt datalehtistä eikä siinä oltu merkitty kytkentöjä, joka aiheutti hieman päänvaivaa ennen osien juottamista. Huomasin levyä katsoessani, että kytkennät näkyivät kirkasta valoa vasten ja päädyin piirtämään nämä tussilla levyyn itse. Arduinoon kiinnitettävää lautaa päädyttiin käyttämään pitkään testien aikana.



Kuva 8: Koekytkentälaudan käyttöä testeissä



Kuva 9: Arduinolle tarkoitettu koekytkinlauta juotettuine releineen

2.4.3 Liikeanturin käyttöönotto

Vaikka liikeanturissa oli vain 3 johtoa niin hieman hämmennystä herätti johtimien värit sillä musta piuha, jonka olin tottunut olemaan maa olikin signaalipiuha ja valkoinen sitten maa. Mustavalkoinen datalehti ei tätä selvästi indikoinut, mutta onneksi anturin piirikortissa oli pienellä tekstillä merkinnät. Kun anturin virta oli kunnossa, vedettiin signaalipiuha vastuksen kanssa suoraan Arduinolle, jota kautta mikro-ohjain saa tiedon havaitseeko anturi liikettä. Koodinpätkä, jolla Arduinolla signaali luetaan ja sen printtaaminen sarjaportin kautta on hyvin yksinkertainen, mutta aluksi tuloksen perusteella anturi ei havainnut mitään, kun yhdistin piuhan analogiatuloon. Oletin, että liikeanturi antaisi havaitsemansa liikkeen analogisena signaalina ja tälle voisi sitten kätevästi asettaa rajaarvon koodissa minkä perusteella laite tulkitsisi, että on puhdistus turvallista tehdä. Verkkohakujen kautta en löytänyt selvää syytä miksi analogiatulo ei tunnistaisi mitään, mutta löysin jonkinlaisesta liikeanturista kytkentäkaavion missä

signaali oli pistettykin digitaalituloon. Piuhan paikan ja koodissa lukuportin korjaamalla sainkin Arduinon kertomaan, havaitseeko anturi liikettä ja jännitemittarilla todennettua, että signaali näyttää jännitteen joko nollassa tai 5 voltissa eli se ei ole analoginen. Tutkin myöhemmin asiaa hieman enemmän koska en ymmärtänyt miksi analoginen portti ei toiminut laisinkaan, vaikka Arduinossa kyllä pystyy muuttamaan analogisia signaaleja digitaalisiksi ja toisinpäin. Virhe ilmeisesti kävi koodini puolella määrittäessäni tuloja, joka on hieman sekavaa Arduinossa. Luin siis väärän portin tietoja ensiksi. Esimerkkinä Arduinolle määritettäessä analogia tuloksi porttia 1 kirjoitin analogRead(1), joka viittaakin porttiin A1. Myös analogRead(15) viittaisi porttiin A1. digitalRead(1) viittaisi sitten taas johdonmukaisesti porttiin 1 [5].

Kun signaali saatiin luettua anturilta niin eteen tuli odotettu ongelma kelluvista signaaleista. Eli tulo, joka ei ole yhdistetty jännitteeseen tai maahan arpoo itselleen arvoja, vaikka koodissa sille on käskytetty olemaan joko LOW tai HIGH. Tätä havainnoin lukemalla tuloa tiheällä taajuudella ja vaikutus oli selvin, kun heiluttelin johtoja tai jopa kun vein käden lähelle tuloa. Uskoin aluksi, että ongelma olisikin liikeanturin virtalähteessä, josta luin yleismittarilla jännitteen olevan vain 10,5 V vaikka 12 V olisi ollut vaadittu. Virtalähteessä oli kuitenkin pieni säätöruuvi, jolla jännitettä hallittiin ja jota kääntämällä asetin sen oikeaan arvoonsa, joka ei ongelmaa kuitenkaan korjannut. Signaalin luotettava lukeminen vaatii, että tulo on kytketty riippuen käyttötarkoituksesta ylös- tai alasvetovastuksella piiriin. Projektin kannalta vastuksen tarkalla koolla ei ole hurjan paljon merkitystä ja päädyin yhden kilo-ohmin alasvetovastukseen, koska liikeanturin signaali tunnistuksessa oli bitti 1. Vastuksen tulisi olla alle kymmenesosan tulon vastukseen verrattuna ja Arduinon dokumentaation mukaan 10 kilo-ohmia olisi ideaali. Kokeilin toimintaa yhden kilon vastuksella, joka on vähän alimitoitettu tähän ja kuluttaa sitten tarpeettomasti hieman virtaa mutta muuten toimisi kyllä

[6]. Vastus korjasi kelluvan signaalin ongelman ja kaikki näytti liikeanturin osalta hyvältä omalla kokoonpanolla.

Kuva 10: Luettu bitti vaihtelee vaikka määritelty alussa tietyksi.

Anturin asennuksessa varsinaiseen laitteeseen tuli vielä pieni haaste sillä tulosta ajoittain mitattiin vain vajaa kolme volttia, kun korkea bitin tunnistus vaatisi yli kolmen. Eli anturi tunnisti oikein, mutta Arduino ei tulkinnut sitä liikkeeksi ja mitään ei tapahtunut. Ongelma selvisi, kun tajuttiin ettei Liikeanturin 12 voltin

virtalähteen maata oltu yhdistetty Arduinon maahan joissa oli yleismittarilla mitattuna pieni potentiaaliero. Virtalähteen ja signaalin välinen ero oli oikein 5 volttia mutta jäi liian alhaiseksi Arduinon maahan verrattuna. Yhdistämällä maat ongelma ratkesi. Maita ei oltu yhdistetty omassa kokoonpanossanikaan, mutta sattumalta siellä ei vastaavaa potentiaalieroa ollut eikä vika tullut sitten testeissä ilmi.

2.4.4 Ohjelmointi

Kun liikeanturi oli saatu toimimaan, siirryttiin valtaosin ohjelmointipuolelle suunnittelemaan, miten liikeanturin tiedon perusteella ohjataan releillä kalvomoottoria ja aktivoidaan valoa. Johdotuksen osalta homma oli yksinkertainen yhden releen ohjatessa kalvomoottoria ja toisen ohjatessa virtaa vahvemmalle ajastinreleelle, jonka läpi verkkovirta ajettiin valolle. Releiden käskyttäminen päälle ja pois oli myös helppoa vaatien vain if-else-komentoja, joissa katsottiin liikeanturin signaali. Haastavammaksi osoittautui pyydetty logiikka, jossa haluttiin ajaa moottoria vasta kun liikkeen havaitsemisesta on kulunut jonkin aikaa, sekä pysäyttää se, jos liikettä havaitaan ajon aikana. Kokeilin ensin yksinkertaisella delay-komennolla, joka näennäisesti toimii, mutta testailtuani ymmärsin komennon pysäyttävän ohjelman kokonaan asetetuksi ajaksi. Tämä siis tarkoittaa, että Arduino ei lue liikeanturiakaan delayn aikana saatikka osaa ajaa muita toimintojaan. Verkosta apua etsiessäni törmäsin neuvoihin käyttää millis()-komentoa. Arduinossa komento käytännössä kertoo, montako millisekuntia on kulunut siitä, kun Arduino on käynnistetty. Tallentamalla luvun muuttujalle ja vertaamalla muuttujaa takasin millikseen, saadaan ohitettua delay-komennon ongelma ohjelman pysähtymisessä. Millikselläonnistuin ajamaan moottoria tietyn aikaa liikeanturin havainnon jälkeen sekä käynnistämään ja sammuttamaan valon moottorin ajon perusteella. Ainoa huono puoli milliksessä, jonka löysin, on se, että milliksen arvo vuotaa ylitse eli alkaa taas nollasta reilu 49 päivän kuluttua

Arduinon käynnistyksestä [7]. Ongelma on sivuutettavissa esimerkiksi millikseen sidotulla uudelleenkäynnistyskomennolla esimerkiksi viikon välein.

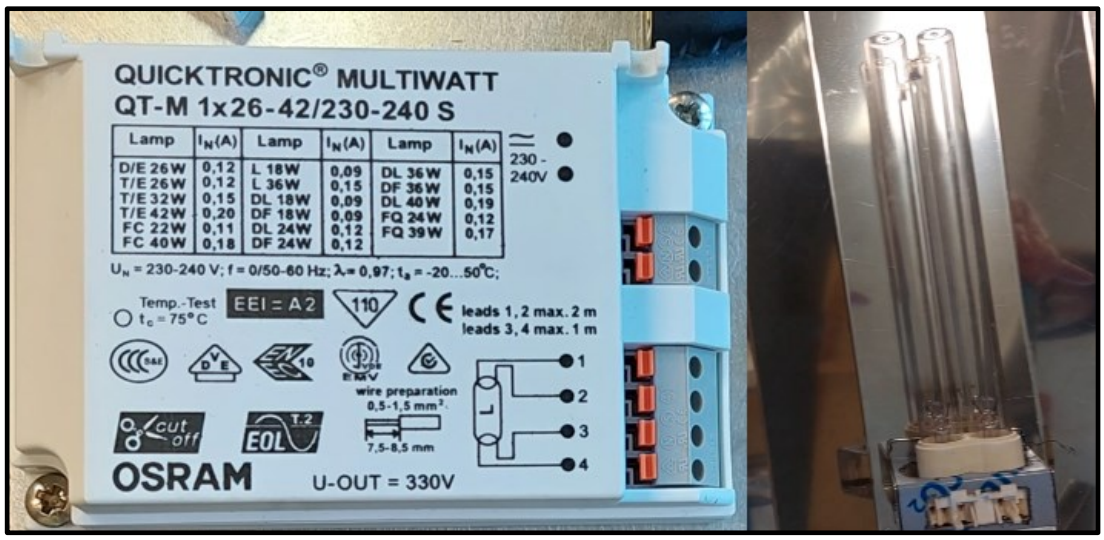
```
36  void liike(){/*Liikesensoritesti*/
37  if (liikesumma <= 20) {/*Liikesensoritesti*/
38    liikeMillis = millis(); //aika resettaa kun liikeanturi aktivoituu
39    digitalWrite(LED_BUILTIN, HIGH);
40    digitalWrite(2, LOW);
41  }
42  if ((startMillis - liikeMillis > 10000) && (startMillis - liikeMillis < 16000)){
43    digitalWrite(LED_BUILTIN, LOW);
44    digitalWrite(2, HIGH);
45    moottori = true;
46  }
47  else { //Lopetetaan ajo
48    digitalWrite(2, LOW);
49    moottori = false;
50  }
51  }
```

Kuva 11: Millis()-komennon käyttö Arduinossa

2.4.5 Valo

Valon ohjaaminen ei vaatinut uusia asioita koodin kanssa sillä kyse oli periaatteessa vaan toisesta releestä liitettynä Arduinoon, joka kytki ajastinreleen kautta valon päälle. Työturvallisuutta sivuttiin tässä yhteydessä varoituksin, että bakteerien tappamiseen tarkoitettu valo on varsin haitallista myös iholle ja silmille, joten valo oli koteloitu aina testikäytössä niin yritystiloissa kuin kotitoimistollakin. Valon releen ohjaamiseen asti olin käytännössä kirjoittanut koodin vain yhteen toistuvaan funktioon, joka minulle epäselvästä syystä ei sitten toiminut enää laajetessaan. Esimerkiksi perus if-else komento ei nähdäkseni toiminut kaikissa tilanteissa johdonmukaisesti kuten olin ainakin uskoakseni sen kirjoittanut, jos eri komentoja oli neljä tai enemmän. Kokeilin myös hetken aikaan case- ja while-komennoilla pyörtää ongelman, mutta en onnistunut. Lopulta päädyin kirjoittamaan koodin vähän uusiksi ja omasta mielestäni ainakin paremmaksi pistämällä loop-komentoni ajamaan erillisiä pienempiä ja hallitumpia funktioita. Tämä ratkaisi ongelmat ja teki koodista muutenkin luettavamman, joka rohkaisi myös yrityksen toista noviisikoodaria tekemään pieniä muutoksia testien aikana, kun tarve sitä vaati. Releen kautta aktivoitiin ajastinrele, joka oli säädetty toimimaan

käytännössä ilman ajastustoimintaansa, mutta oli sen verran korkeaksi mitoitettu, että voitiin johtaa valon tarvitsema virta Quicktronic multiwatt liitäntälaitteelle. Liitäntälaitteen tarkoitus on suojella lamppua toistuvien sytytysten ja sammutusten aikana ja nostaa verkkovirran 230V jännite lampulle sopivaksi 330V.

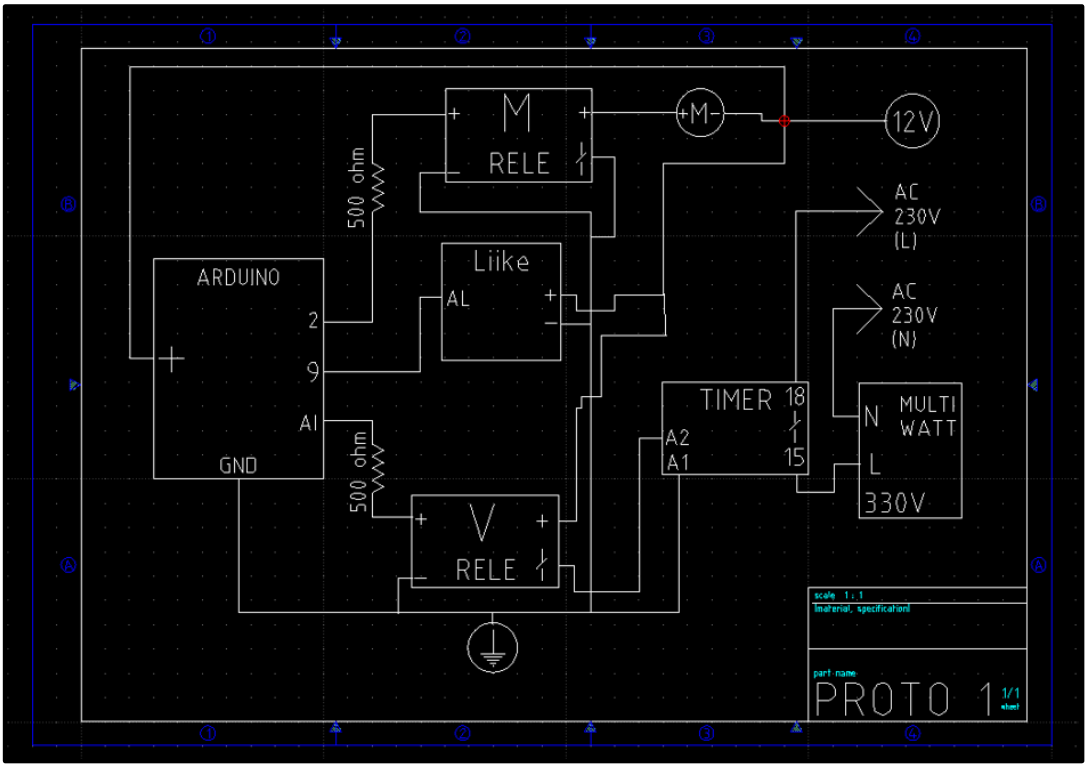


Kuva 12: Liitäntälaite ja valo

2.4.6 Loki ja prototyypin esittely

Törmäsin vielä pieniin haasteisiin kirjoittaessani Arduinon näkemiä lukuja sarjaportinkautta ohjelmointiympäristöön, jota olin käyttänyt koko kehityksen ajan. Ohjelmoin Arduinon kertomaan jatkuvasti kaikista tuloista tietoa, joiden kautta ymmärsin nopeammin ongelmatilanteissa missä vika voisi olla. Esimerkiksi Arduino kertoi, onko liikeanturi havainnut mitään bitillä yksi, ja jos ei niin bitillä nolla. Tulotietojen, ykkösten ja nollien sijaan laite palautteli satunnaisesti kirjaimia ja erikoismerkkejä. Aluksi pelästyin, että olenko rikkonut jotain joko Arduinosta tai komponenteista. Yleismittarilla katsottuna kaikki kuitenkin toimi kuten pitää ja tajusin vian olevan vain tietojen kirjauksessa. Eli lähetin tietoja liian nopeasti tai jopa päällekkäin, jolloin tiedot eivät tulleet oikein perille. Ominaisuus johtuu siitä, että Arduinon datan lähetys ja vastaanotto on epäsynkroninen eikä siis ole sidottu kellosignaaliin joka huolehtisi viestien ketjuttamisesta automaatti-

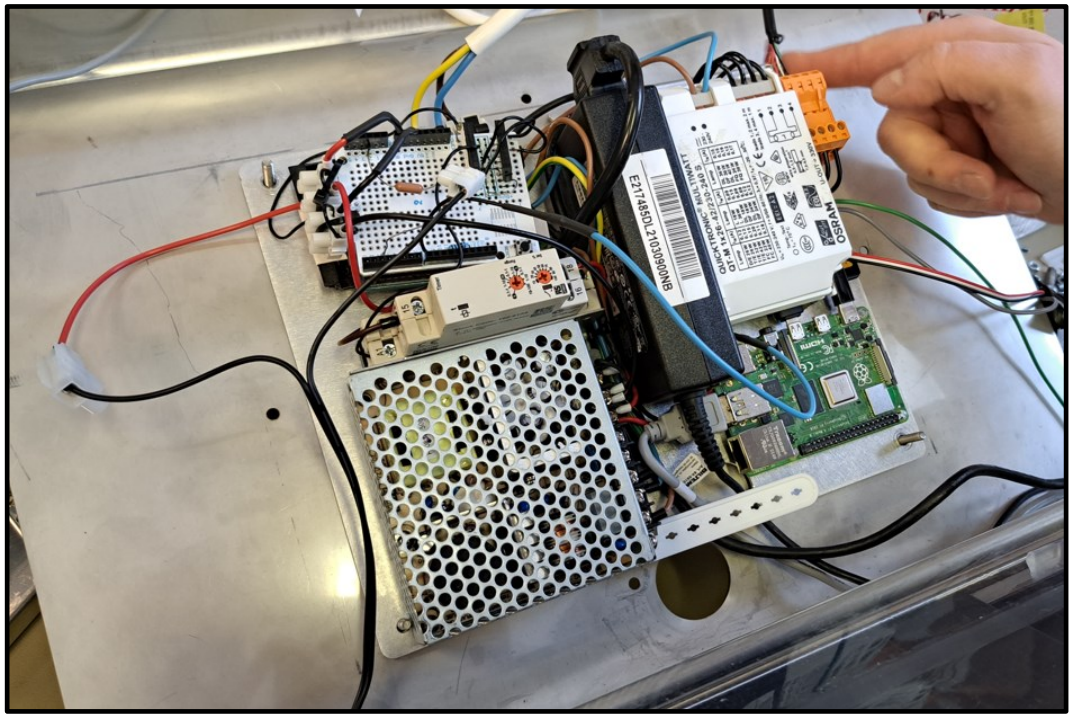
sesti [8]. Ratkaisin tämän ongelman hyvin lyhyillä delay-komennoilla, joilla tauotin, kuinka usein tietoja lähetetään. Millis-komento olisi tähänkin varmasti parempi vaihtoehto, mutta kommunikointikoodin kirjoittaminen milliksen kanssa olisi varmaan tuplannut koodin pituuden eikä vastaavaa logia oltu enää pyydetty lopulliseen sovellukseen joten en vaivautunut sitä tekemään. Arduinolle olisi myös ollut komento, jolla tarkistetaan onko sarjaportti auki ennen kuin sen kautta kokeillaan lähettää lisää dataa joka olisi voinut olla yksinkertainen ratkaisu.



kuva 13: Kollegan puhtaaksi piirtämä kytkentäkaavio proto 1:stä

Ensimmäisen prototyypin lopuksi saatiin kaikki komponentit asennettua suhteellisen vaivattomasti tarkoitettuun koteloon. Tila oli hyvin ahdas johtuen useammasta piirikortista ja eri virtalähteistä, joten johdottaminen ja asennus vaati hieman sorminäppäryyttä. Samoin erityisesti Arduinon eristäminen niin metallisen kotelon pohjasta kuin koekytkinlevystä oli haastavaa ja tämä tehtiin pajalta löytyneistä muovinkappaleista sekä eristysteipistä. Sekavasta ulkoasustaan huoli-

matta laite kuitenkin toimi sillä tasolla, että sitä kehdattiin esitellä yrityksen ulkopuolellakin. Käsittääkseni vastaanotto oli positiivinen, mutta en itse esittelyihin osallistunut vaan jatkettiin kehitystä seuraavaan vaiheeseen.



Kuva 14: Ensimmäinen prototyyppi kasattuna koteloalustalleen

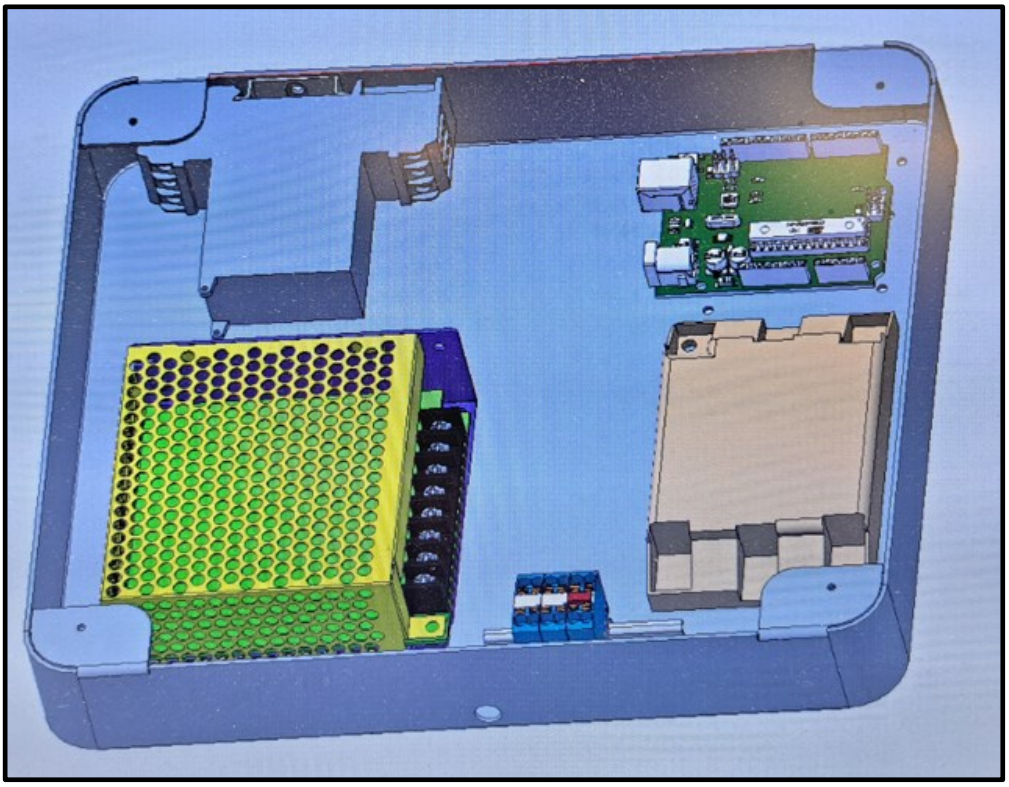
2.5 Prototyypin jatkokehitys

Prototyyppilaitteen versio 2:ssa tehtiin jatkokehitystä laitteen mekaniikkaan, elektroniikkaan sekä kotelointiin. Merkittävimpiä uudistuksia olivat näytön uusiminen, jossa vaihdettiin versio 1 käytössä ollut erillinen Rasberry PI:n pyörittämä linux järjestelmä valmiiseen näytön sisälle asetettuun tietokoneeseen. Toiminnallisuus pysyi samana, mutta Rasberryn poistaminen antoi huomattavasti lisätilaa muulle elektroniikalle kotelossa. Lisätila antoi vähän mukavammin liikkumavaraa muiden komponenttien asettelussa, mutta varmastikin merkittävämmin vaikuttaa lopullisen kotelon kokoon, joka halutaan pitää siistinä ja tehokkaana. Rasberry olisi muuten varmasti soveltunut tarkoitukseen varmasti ihan hyvin, sillä sen suorituskyky ja mahdolliset toiminnallisuudet olivat käsittääkseni huomattavasti paljon halvempaa Arduinoa suuremmat, mutta nähtiin kuitenkin

mielekkääksi jatkaa Arduinolla koska se oli minulle tutumpi sekä meillä oli Arduinoja valmiiksi useampia. Arduino, tai mikro-ohjaimet yleensäkin, eivät sovellu suurten jännitteiden tai virtojen käsittelyyn ja Arduino esimerkiksi kestää maksimissaan 20mA virtaa per lähtö [9]. Edellisesti Arduino projektissa taisin saada puolenkymmentä lähtöä toimimattomiksi erinäisten transistori- tai relejohdotusten ollessa hieman pielessä niin tämänkin vuoksi puolsin, että käytetään näitä halvempia osia niin virheet eivät sitten kirpaise niin paljoa. Jälkikäteen voin todeta, että ehkä olin hieman pessimistinen osaamiseni suhteen, sillä mitään en tietääkseni saanut rikki koko projektin aikana.

2.5.1 Kahdentaminen ja liikeanturin vaihto

Useampi Arduino mahdollisti työskentelyn useammassa kohteessa, sillä pystyin yksittäisiä toiminnallisuuksia kokeilemaan koodissa kotona ja sitten vain lähettämään päivitetyn koodinpätkän yritykselle. Kollega syötti koodin protolaitteessa olevaan Arduinoon ja katsoi, toimiiko ohjelma käytännössä osana varsinaista kokonaisuutta. Jos olisin aina joutunut mennä itse paikan päälle kokeilemaan muutoksia olisi päivätyön jälkeinen ilta venähtänyt varsin pitkäksi. Kävin oikeastaan paikalla vain, kun tehtiin isompi muutos tai vaihdettiin jotakin komponenteista uuteen, joka vaati erilaista johdotusta tai vähän passiivisten komponenttien asettelua eri paikkoihin. Pääasiassa tällöin työskentelimme vain sunnuntaisin, ellei ollut valtava kiire varmistaa toiminta tai löytää johonkin ongelmaan ratkaisu.



Kuva 15. Versio 2 suunnitelma kotelosta

Liikeanturi päätettiin vaihtaa toisen valmistajan vastaavaan PIR-anturiin. Uuden anturin kytkentä oli hieman yksinkertaisempaa, koska tämä toimii vain 5V jännitteellä verrattuna vanhaan. Saatiin siis virta suoraan Arduinolta. Johtojen määrä pysyy samana, mutta koska kaikki muutkin anturin piuhat joutuvat menemään Arduinolle niin johdotus on siistimpää. Anturi muutti hieman ohjelmointia sillä vähän yllättävästi toimi käänteisellä logiikalla vanhaan verrattuna. Anturin tunnistaessa siis jännite signaalijohtimessa nousi eikä laskenut. Tämä onneksi selvisi nopeasti yleismittarilla, jonka jälkeen koodin muuttaminen sopivaksi oli muutaman minuutin homman. Alasvetovastus piti myös vaihtaa ylösvetovastukseen, mutta kätevästi tälläinen toiminnallisuus on jo Arduinossa valmiiksi ja vastuksen saa aktivoitua lisäämällä komennon tulon määrittelyyn koodissa. Uusi liikeanturi toimi huomattavasti herkemmin kuin vanha. Pöydällä kasattuna anturi tunnisti liikkeen, kun kävin noin kahdeksan metrin päässä olevasta ovenpielestä vain kurkistamassa. Vanhemmalla anturilla tuntui tulevan raja vastaan parin metrin kohdalla, joten ero oli merkittävä. Mietimme tunnistaako uusi anturi jopa liian

herkästi käyttötarkoitusta varten, sillä jokaisen etäisemmänkin ohikulkijan tulkinta päätteen käyttäjäksi jättäisi puhdistusten taajuuden mahdollisesti aika alhaiseksi. Liikeanturi on kuitenkin sijoitettu alaviistoon prototyyppilaitteessa, joka sokeuttaa anturia kauempana tapahtuvalle liikkeelle. Lisäksi ehdotin, että anturille voisi myös muuttaa tai lisätä kotelointia linssin osalta, sillä infrapuna-anturreihin on saatavilla edullisesti erilaisia linssejä jotka vaikuttavat herkkyyteen paksuuden ja linssin muodon mukaan [10].

Kuva 16: Liikeanturin sijainti laitteessa

2.5.2 Liiketiedon koodauksen kehittäminen

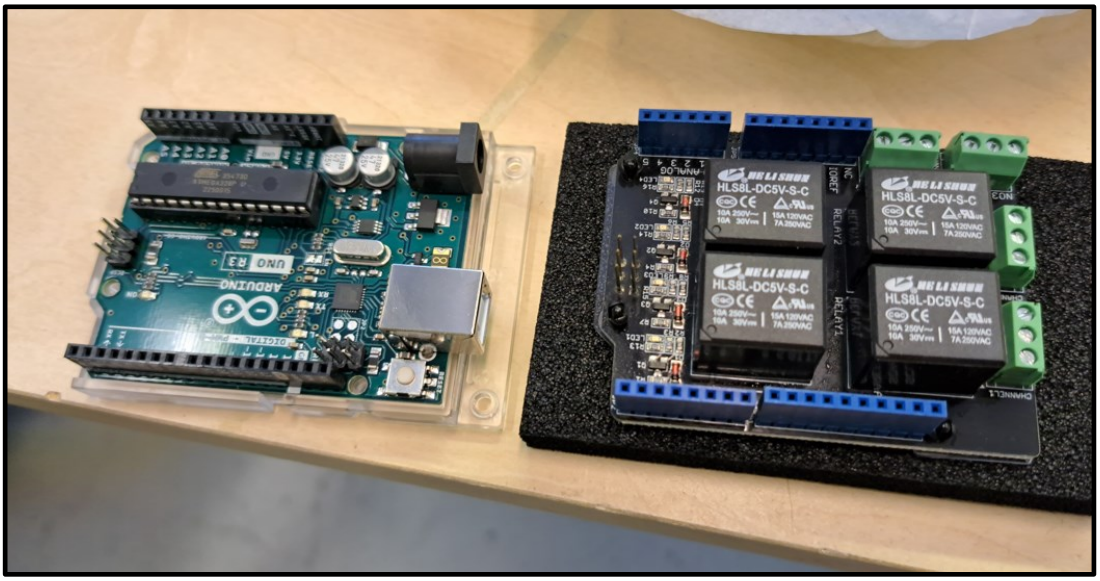
Suurimmilta osin vanhan liikeanturin vaikeuksien takia olin jo aiemmin aloittanut pohtimaan ja kokeilemaan miten liikeanturin tunnistustiedon saisi tehtyä luotettavammaksi. Päätin, että koska anturi palauttaa liiketietoa hyvin nopeasti niin voisimme kokeilla tulkita liikkeen vasta useammasta havainnosta. Kokeilin ensin kirjoittaa koodin siten, että vaaditaan tarpeeksi monta perättäistä liikehavaintoa, ennen kuin liike tunnistetaan ihmiseksi. Tämä oli hieman parempi, mutta vieläkin tuli haasteita koska ajoittain anturi ei tulkinnut liikettä aivan joka lukukerralla, kun kävelimme päätteelle ja luimme lokitietoja samalla. Eli oli vieläkin mahdollista, ettei laite havaitsisi käyttäjää ja alkaisi esimerkiksi tekemään puhdistusta samalla kun henkilö yrittää päätettä käyttää. Seuraavaksi kokeilin summata lukutuloksia eli kirjoitin for-luupin katsomaan 100 kertaa 5 ms välein näkeekö liikeanturi mitään. Jokaisen lukukerran tulos summataan edelliseen tulokseen, jolloin luupin lopuksi tulos olisi esimerkiksi 30 tarkoittaen että 30 kertaa 100:sta lukukerrasta havaitsimme liikettä. Tulos tallennetaan toiselle muuttujalle luupin sisällä ja sitä verrataan asetettuun arvoon, esimerkiksi suurempi kuin 50:tä, joka sitten tulkitaan liikkeeksi. Luupin lopuksi kaikki muuttujat nollataan ja luuppi alkaa alusta. Tällä onnistuimme tekemään anturitiedoista tarpeeksi luotettavan ja herkkyyttä on myös koodin puolella helppo säätää vain vaihtamalla vertailuarvoa. Ratkaisu yllätti vähän yksinkertaisuudellaan ja ainoa haaste oli jälleen omat virheet koodinkirjoituksessa, kun menin sekaisin merkkien = ja == välillä. =-merkki tarkoittaa, että muuttuja asetetaan määritellyksi ja ==-merkki taas, että muuttujaa verrataan määriteltyyn.

```
void lukeminen() {
for (luku; luku < 100; luku++){ //Monestako lukutuloksesta liikkeen havaitseminen lasketaan
  summa += digitalRead(2); // A += B. is the same as: A = A + B
  delay(5);
}
liikesumma = summa; //Tallennetaan summa josta päätellään nähtiinkö mitään
Serial.print("Liikesumma ");
Serial.print(liikesumma);
Serial.println();
if (liikesumma <= 50){ // Monestako lukutulosta tulkitaan liikkeeksi
  digitalWrite(LED_BUILTIN, HIGH);
}
else{
digitalWrite(LED_BUILTIN, LOW);
}
luku = 0;
summa = 0;
}
```

Kuva 17: Liikeanturin lukutulosten summailua koodissa

2.5.3 Koekytkentälevyn korvaaminen

Päätettiin myös etsiä helpompaa ratkaisua itsejuotetulle koekytkentälevylle releineen ja vastuksineen. Oma tekemä toimi kyllä hyvin, mutta on vaivanloinen kahdentaa tai muokata eikä järin siisti, jos sitä tulisi jollekulle esitellä. Kytkentälevyn kanssa olisi tullut myös enemmissä määrin hankaluuksia mahdollisten lisäkomponenttien kanssa sillä tila oli nopeasti loppumassa. Olin aiemmin lukenut, että Arduinolle olisi useanlaisia suojia eli laajennuslevyjä, jotka asetetaan Arduinoon suoraan kiinni ja tuovat mukanaan valmiita lisäominaisuuksia tai komponentteja. Löysimmekin Seeed studion valmistaman relay shieldin. Suojassa on 4 relettä, joita voidaan kytkeä suoraan Arduinon ulostuloilla ja ne ovat mitoitettu kestämään peräti 10A ja 30V eli soveltuvat pienten sähkömoottorien ohjaamiseen. Jokaisella releellä on myös kätevä led-valo näyttämässä milloin mikäkin rele on kytketty, joka auttaa ohjelmoinnissa havainnoimaan toimiiko laite halutulla tavalla ennen kuin muita toimilaitteita yhdistetään releisiin. Huonona puolena verrattuna vanhoihin releisiin suuremmat mekaaniset releet päästävät myös huomattavasti lujemman äänen kytkeytyessään ja ovat muutenkin tarpeettoman ylimitoitettuja loppusovellukselle.

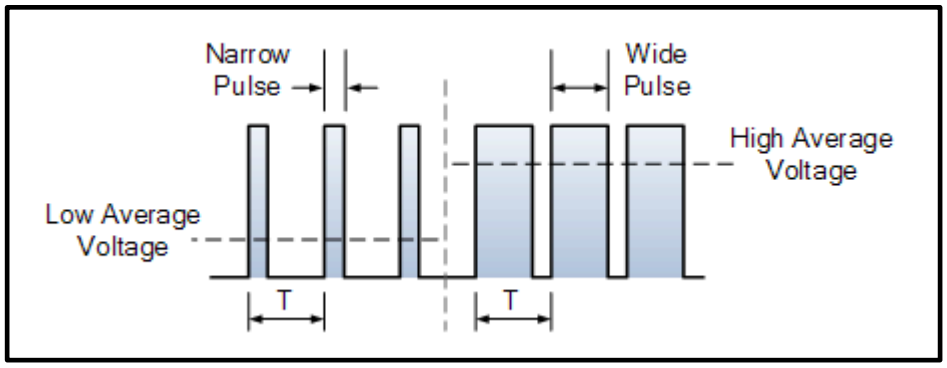


Kuva 18: Arduino ja relesuoja

2.5.4 Kalvomoottorin hallittu ohjaaminen

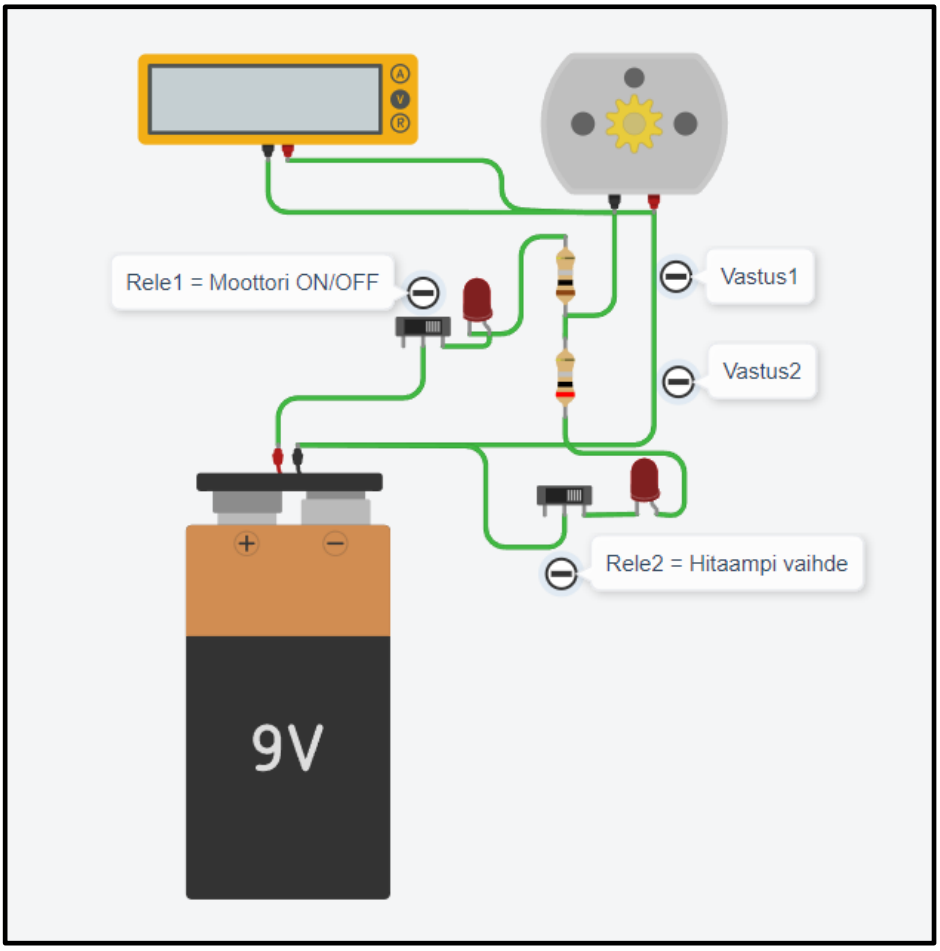
Kalvoa pyörittävään moottoriin toivottiin parannuksia. Moottorin kiihtyvyyden kanssa oli haasteita sillä kalvoa pyörittävä moottori suti kalvon alla, joka vaikutti niin kalvon kulumiseenkin kuin siihen mihin kohtaan kalvo ajetaan, koska moottori toimi vain kovakoodatulla ajastuksella. Toivottiin, saataisiinko moottorille jonkinlainen ramppi ja nopealla tutkiskelulla on ihan mahdollista. Minulle tarjottiin ensiksi enkooderia, että onnistuisiko ramppi pelkästään sillä, mutta muistelin ja myöhemmin varmistin, että enkooderi vain kertoo tarkasti asentoaan eikä itsessään voi hallita moottorin liikettä [11]. Enkooderin asentotietoon perustuen ohjauksen pystyisi tekemään kuitenkin laadukkaaksi eli moottorin kiihtyvyys sekä aika olisi mahdollista koodata tiedon perusteella ajettavaksi. Moottorin suuntaa voisi ohjata H-sillan avulla, johon on valmiita komponentteja ja näyttää teoriassa suhteellisen yksinkertaiselta vaikkakin koodauksen osalta varmaankin menisi paljon aikaa ja opettelua. H-sillalla on myös väärin ohjattuna hyvin helppoa saada oikosulku aikaiseksi, sillä se koostuu yksikertaisimmillaan neljästä transistorista, jotka on kytketty moottorin napojen ympärille [12]. Pohdin myös PWM (Pulse Width Modulation, pulssileveysmodulaatio) käyttämistä moottorin nopeuden hallitsemiseen. PWM avulla jännitetasoa vaihdellaan korkean ja matalan välillä nopealla taajuudella. Muuttamalla pulssisuhdetta voidaan rajoittaa,

paljonko virtaa syötetään moottorille ja moottorin nopeus muuttuu. Arduinolla saisi moottoria ohjaavan releen lähdön helposti muutettua käyttämään PWM-toiminnallisuutta, mutta pelkäsin, että tämä rasittaisi relettä sitten liiaksi ja lyhentäisi kohtuuttomasti sen käyttöikää. Luin vielä jälkikäteen, että vaikka PWN yhdistämistä releeseen ei juurikaan suositella, niin juuri prototyyppi 1:ssä käytettyjen SSR-releiden kanssa voisi tämäkin toimia sillä niissä ei ole mekaanisia osia, jotka kuluisivat nopeasti loppuun.



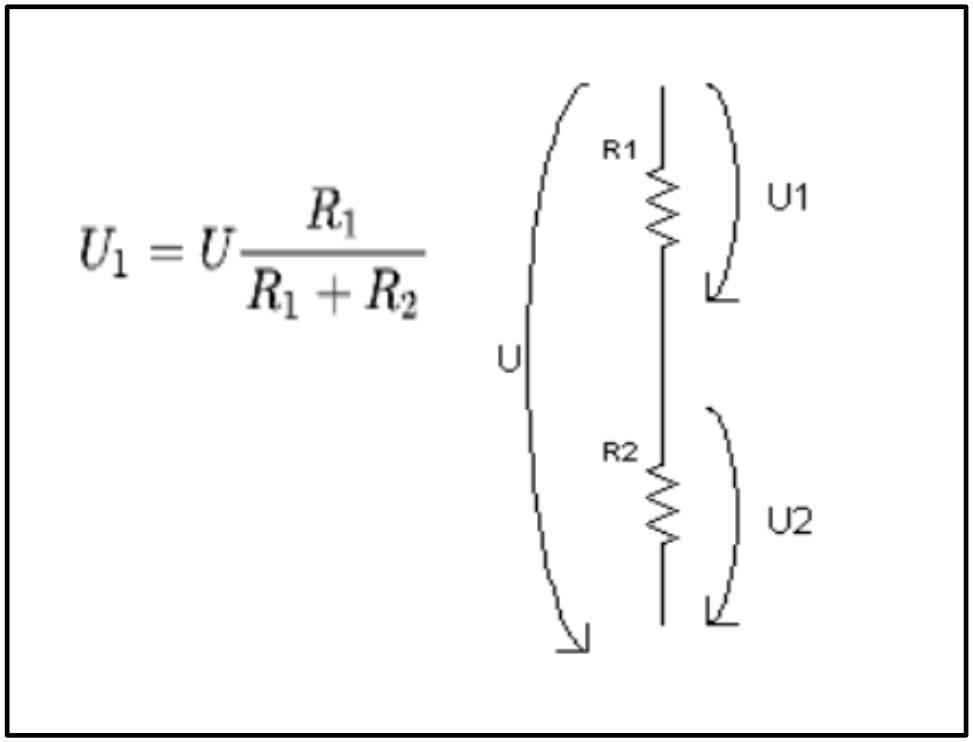
Kuva 19: Pulssileveysmodulaatio [13].

Ehdotin lopulta ratkaisuksi, että kokeiltaisiin jännitteenjakajaa uuden relekilven ylimääräisten releiden kanssa niin meidän ei tarvitsisi hankkia uusia komponentteja ja keksiä niille sijaintia sovelluksessa.



Kuva 20: Jännitteenjakajan simulointi

Simuloin ensin tinkercadilla, että jännitteenjakaja toimisi kuten ajattelin ja käytin simulointia esitelläkseni idean. Jännitteenjakajalla voisimme ajaa ensin sekunnin tai kaksi kalvon moottoria esimerkiksi kahdeksalla voltilla ja katkaisemalla sitten releellä maayhteyden jälkimmäisestä vastuksesta, jolloin moottori ajettaisiin täydellä 12 voltilla. Olin hieman huolissani rajoittaako vastusten lisääminen moottorin nopeutta liikaa, sillä oletin, ettei sähkömoottorilla olisi itsessään juurikaan vastusta. Mittasin vastuksen sähkömoottorin yli ja tulos oli lähemmäs 1000 ohmia, joka helpotti vastusten mitoittamista. Idea toimi myös kasattuna hyvin, kun kytkin nopeasti mekaanisen kytkimen ajamaan jakajareleen virkaa. Nappia painamalla moottori hidastui ja napin vapauttamalla palasi täyteen ajonopeuteen.



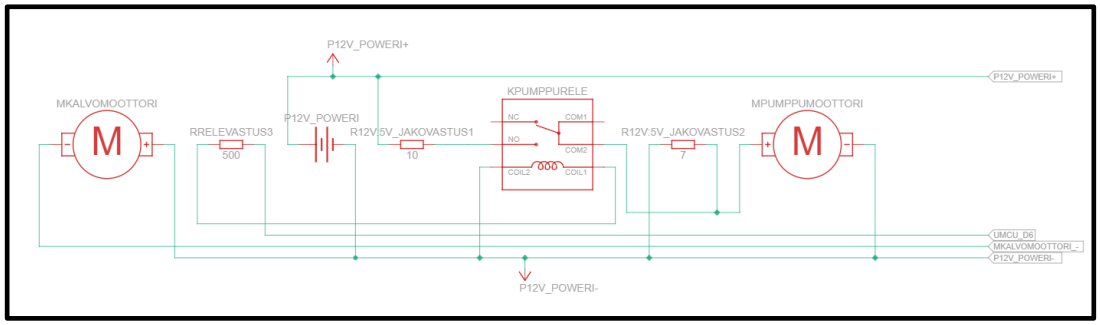
Kuva 21: Jännitteenjakajan laskentakavaa [14].

Kalvon ajamista oikeaan tai toistettavaan asentoon enkooderin lisäksi voisi olla yksinkertaisempaa käyttää jonkinlaisia valoverhoantureita. Valoverhojen toimintaperiaate on, että anturin yksi osa lähettää valosädettä vastaanottavalle osalle, joka osaa kertoa, kun säde katkeaa. Jos kalvoon saisi piirrettyä tai teipattua valoverhon katkaisevan kohdan, tunnistaisimme, milloin käytetty osa kalvosta on saatu ajettua puhdistukseen ja puhdistettu osa käyttöön. Haastavaksi logiikan kannalta voisi tulla tarvittava ominaisuus moottorin kytkemisestä pois päältä, mikäli henkilö kävelee näyttöpäätteelle kesken kalvonsiirron. Uskoisin, että jos tämä todetaan ongelmaksi niin tilanteelle pystyisi kirjoittamaan oman koodinpätkänsä.

2.5.5 Mekaanisen puhdistuksen suunnitelmia

Valon avulla todistetusti kalvon puhdistaminen bakteereista on tehokasta, mutta suunnitelmissa on myös lisätä mekaanisempi puhdistus tahrojen putsaamiseksi kalvosta. Alustavasti tähän on ajateltu sähkömoottoreilla pyörivää huoparullaa ja pumppua. Molempia ohjattaisiin mikro-ohjaimella vastaavalla tavalla kuin mitä

olemassa olevia toiminnallisuuksia ohjataan, jolloin huoparulla pyörisi suurin piirtein samaan aikaan kun kalvoa pyöritetään hangaten kalvon pintaa. Huoparullaa kostutettaisiin 5 voltin sähkömoottorilla toimivalla pumpulla, joka ruiskuttaisi vähän esimerkiksi isopropanolia. Isopropanolia käytetään yleisesti puhdistamiseen sen liuottavien ominaisuuksien takia ja aine myös haihtuu nopeasti huoneenlämmössä, joten pinta ei jää märäksi [15]. Koska 5 Voltin lähdettä ei suoraan nykyisessä laitteessa ole, joka soveltuisi moottorin pyörittämiseen, otamme jännitteenjakajalla virran 12 Voltin lähteestä moottoria ohjaavan releen yhteydessä.



Kuva 22: Kalvo- ja pumppumoottoreiden sähköpiirustus Simulink-ohjelmasta

3 Lopputulos

Työn aikana suunniteltiin ja kasattiin näyttöpääte sekä elektroniikka. Näytön ympärillä olevaa kalvoa pyöritetään liiketunnistusanturin lukeman perusteella. Mikro-ohjaimessa oleva ohjelma ohjaa releitä, joiden kautta moottorilla näytön päällä oleva osa kalvosta ajetaan näytön taakse. Näytön takana oleva lamppu puhdistaa bakteerit kalvosta myös mikro-ohjaimen ohjelmalla ja releillä. Toiminnot keskeytetään, mikäli liikeanturi havaitsee liikettä kesken puhdistuksen, jotta laite on turvallinen käyttää. Ohjelmaan kirjoitettu koodi on kommentoitu ja palauttaa myös ohjaimen eri tulojen ja lähtöjen arvoja, joka helpottaa jatkokehitystä nykyisellä ohjaimella. Liikeanturin koodi on kirjoitettu, siten että herkkyyttä voidaan säätää tarpeen mukaan yhdellä parametrilla, jos anturi vaihdetaan tai sijaintia muutetaan. Elektroniikasta on tuotettu kytkentäkaavio, jossa kaikki komponentit on nimetty käyttötarkoituksensa mukaan, jotta jatkokehitys olisi

mahdollisimman sujuvaa. Laitetta on saatu esiteltyä kiinnostuneille asiakkaille tässä muodossaan.

Työn tavoitteita ei saavutettu. Alkuperäinen suunniteltu lopputulema piti sisällään varsinaisen kaupalliseksikin soveltuvan piirikortin, jossa toimintoja ohjattaisiin muulla kuin protoiluun tarkoitetulla mikro-ohjaimella, mahdollisesti ohjelmoitavalla logiikalla. Kehitystyössä törmättiin useisiin haasteisiin, joista suurimpia olivat opinnäytetyön aiheen ulkopuolella oleva soveltuvan kalvomateriaalin löytäminen, sekä epävarma rahoitustilanne. Myös näyttöpäätteen ja elektroniikan kotelointi vei paljon aikaa. Omalta osalta liikeanturin tunnistusongelmat, jotka tuntuvat vaivaavan vasta kun anturi kiinnitetään laitteeseen, vaati paljon ihmettelyä. Myös vaaditut toiminnallisuudet ovat eläneet ja lisääntyneet työn aikana sitä mukaan testikäyttö on edennyt ja ongelmia ilmennyt.

Lähteet

1 RS PRO SPDT Multi Function Multi timer -, ON Delay, DIN Rail. PDF-dokumentti. Saatavissa: https://docs.rs-online.com/f871/A700000008913496.pdf. [Viitattu 10.3.2025]

2 Ada, L. How PIRs Work. WWW-Dokumentti. 2014. Saatavissa: https://learn.adafruit.com/pir-passive-infrared-proximity-motion-sensor/how-pirs-work. [Viitattu 10.3.2025]

3 SE-10. PDF-Dokumentti. Saatavissa: https://www.pololu.com/file/0J250/SE-10.pdf. [Viitattu 10.3.2025]

4 Autodesk Tinkercad. WWW-Dokumentti. Saatavissa: https://www.tinkercad.com/. [Viitattu 10.3.2025]

5 Arduino Stack Exchange. WWW-Dokumentti. 2015. Saatavissa: https://arduino.stackexchange.com/questions/13522/can-i-use-analogread-to-read-a-digital-pin. [Viitattu 10.3.2025]

6 Soldered Electronics. What is a pull-up / pull-down resistor. WWW-Dokumentti. 2024. Saatavissa: https://soldered.com/learn/what-is-a-pull-up-pull-down-resistor/. [Viitattu 10.3.2025]

7 Arduino Forum. Using millis() for timing. A beginners guide. WWW-Dokumentti. 2017. Saatavissa: https://forum.arduino.cc/t/using-millis-for-timing-a-beginners-guide/483573. [Viitattu 10.3.2025]

8 Siebeneicher, H. Universal Asynchronous Receiver-Transmitter (UART). Arduino Docs. WWW-Dokumentti. 2024. Saatavissa: https://docs.arduino.cc/learn/communication/uart/. [Viitattu 10.3.2025]

9 Electronic Components Store. Getting Started with Arduino. WWW-Dokumentti. 2018. Saatavissa: https://www3.ntu.edu.sg/home/ehchua/programming/arduino/Arduino.html. [Viitattu 10.3.2025]

10 Murata Manufacturing Co. Basics - What is the Fresnel lens used in infrared sensors? -. WWW-Dokumentti. Saatavissa: https://www.murata.com/products/sensor/infrared/overview/basic/lens. [Viitattu 10.3.2025]

11 Encoder Products Company. The Basics of How an Encoder Works. WWW-Dokumentti. Saatavissa: https://www.encoder.com/wp2011-basics-how-an-encoder-works. [Viitattu 10.3.2025]

12 Hutasu. H-silta. WWW-Dokumentti. 2017. Saatavissa: https://www.hutasu.net/elektroniikka/teoriaa/h-silta/. [Viitattu 10.3.2025]

13 Aspencore. Pulse Width Modulation. WWW-Dokumentti. Saatavissa: https://www.electronics-tutorials.ws/blog/pulse-width-modulation.html. [Viitattu 10.3.2025]

14 Wikipedia. Jännitteen- ja virranjakosääntö. WWW-Dokumentti. 2024. Saatavissa: https://fi.wikipedia.org/wiki/J%C3%A4nnitteen-_ja_virranjakos%C3%A4%C3%A4nt%C3%B6. [Viitattu 10.3.2025]

15 Työterveyslaitos. Isopropanoli. WWW-Dokumentti. 2025. Saatavissa: https://ova.ttl.fi/isopropanoli. [Viitattu 10.3.2025].

Liitteet

Bacterial communities at airport assessed by high-throughput sequencing, VTT, Tsitko, Salo, Kulmala, Mauko, 2018

Background

Air travel enables rapid global transport of infectious diseases. Hub airports visited by hundreds of thousands of passengers a day are potential environments for spreading infectious diseases. Therefore, airports need a functional prevention mechanism and a response plan in case of a disease outbreak. With growing number of passengers, the contaminated surfaces could play an important role in spreading the diseases.

Sufficient understanding of the microbial communities on surfaces, which could come in contact with the passengers, can facilitate the development of rapid, sensitive and specific tools for monitoring the microorganisms of interest.

Methods

Bacterial populations from 44 samples (22 sampling points, sampled twice with an interval of 2 weeks) were characterized by high-throughput sequencing (Ion-Torrent platform) of 16S rRNA gene's V3-V4 variable region. The sequence data were analysed using the Mothur software (Schloss et al., 1999). Sequences were classified using the RDP Bayesian Classifier (80% confidence) and clustered into phenotypes at different phylogenetic levels.

Results

This study reveals diverse bacterial populations present on all surfaces at the airport. A total of

21 bacterial phyla were detected. The main phyla (> 97% of all sequences) were Actinobacteria, Proteobacteria and Firmicutes. The sequences were grouped into 569 phylotypes at genus level.

Bacterial community profiles differed between the sampling places as well as different sampling times. The greatest variation in bacterial flora was found in children's play areas.

Most of the genera were identified to be human associated. In this survey no pathogenic bacterial groups were found.

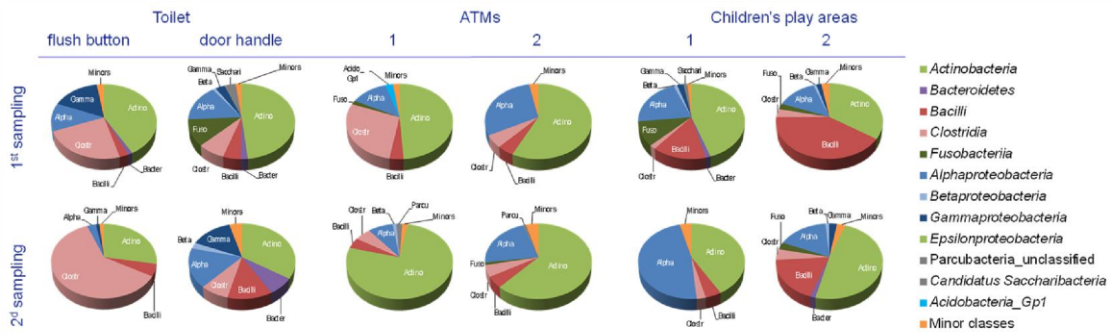


Figure 1. Bacterial population profile at phyla-class level recovered from selected surfaces at airport as an example of diverse bacterial populations.

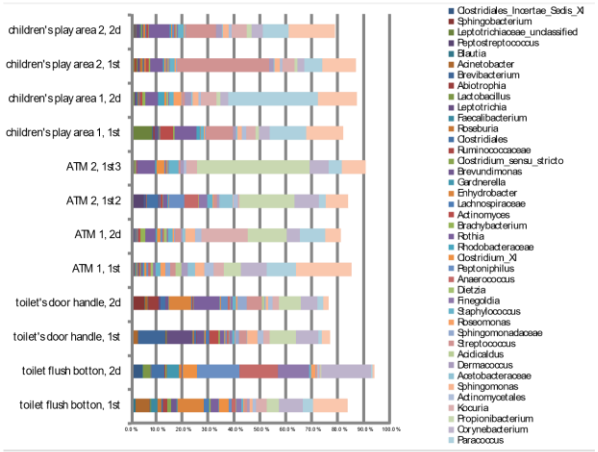


Figure 2. Main bacterial genera or classes identified from the surfaces

Conclusions

High-Throughput Sequencing is a useful tool for mapping the microbial communities over the large number of samples.

The approach can reveal bacterial genera, which are difficult to cultivate and therefore, can be overlooked in conventional microbiological survey.

HTS can be employed as a first line screening to identify the critical point at airport and the results can be used as a background information in risk assessment.

For comprehensive risk assessment more information on passenger's movements, cleaning procedures etc. should be collected.

Acknowledgements

The PANDHUB Project (Prevention and Management of High Threat Pathogen Incidents in Transport Hubs) has received Satu Salo funding from the European Union's Seventh Framework Programme for research, technological development and satu.salo@vtt.fi demonstration under grant agreement no 607433

Contacts

Sähköposti, Peter Christiansen, 2023

Moi,

ClearReino Oy tarina alkaa vuonna 2018 kun VTT julkaisi tutkimuksen: Bacterial communities at airport assessed by high-throughput sequencing. Yhtenä tutkimuksin kriittisitä pinnoista oli kosketusnäytöt.

Tähän ongelmaan suunnittelija Reino Malinen alkoi ideoida UVC-valoon pohjautuvaa ratkaisua heti hygieenisestä kosketusnäytöstä. Tutkittuaan, että vastaavaa näyttöä ei vielä oltu patentoitu päätti hän suunnitella laitteensa valmiiksi. Idean ympärille hän keräsi tiimin kokeneista kumppaneista. Myynnistä ja markkinoinnista vastaavaksi valikoitui Peter Christiansen, joka on vuodesta 2001 lähtien kaupallistanut eri teollisia hygieniaratkaisuja mm. HK-Ruokatalo Oy ja Sinebrychoff Oy. Lopuksi tiimin täydensi pitkäaikainen ohjelmisto ja tuotekehitysyrityksen perustaja ja Pehitec Oy:n toimitusjohtaja Pekka Päivärinta. Tämän tiimin voimin keksintöä vietiin eteenpäin siten, että yhtiö perustettiin 2022. Samalla alkoi keksinnön patentointi, joka saatettiin loppuun vuoden 2023 aikana. Tämän jälkeen patentointia on jatkettu yhteisellä EU patentilla. Proto 1 saatiin valmiiksi keväällä 2023. Yhteistyö VTT:n kanssa alkoi heti tämän jälkeen keväällä 2023 laitteen toiminnan varmistamiseksi. Tulokset yllättivät, jopa tutkijat sanomalla, että ”laite tuhosi pintamikrobit paremmin mitä olisi uskonut”. Tämän jälkeen laitetta on esitetty useille kansainvälisille tahoille. Heti hygieeninen kosketusnäyttö herättää paljon mielenkiintoa ja Proto 2 varten tarvittavaa arvokasta palautetta on saatu. Markkinat odottavat ja kansainvälistyminen voi alkaa 2024.

Peter