PARISTOKÄYTTÖISEN HERÄTYSKELLON LIITTÄMINEN LÄHIVERKKOON

Arttu Afflekt

Ohjaaja: Juha Häkkinen

SÄHKÖTEKNIIKAN TUTKINTO-OHJELMA

2018

Afflekt A.N. (2018) Kandidaatintyö. Oulun yliopisto, Elektroniikan ja tietoliikennetekniikan tutkinto-ohjelma. Kandidaatintyö, 38 s.

TIIVISTELMÄ

Tämä kandidaatin työ käsittelee mekaanisen paristokäyttöisen herätyskellon liittämistä lähiverkkoon, sekä sen tuomia kellon lisäominaisuuksia. Työssä käydään läpi kelloon lisättyjen ominaisuuksien suunnitteluprosessi ja selvitetään niihin liittyvää teknistä taustatietoa. Kellon suunnitellut lisäominaisuudet ovat: herätysajan säätäminen, kesä- ja talviajan vaihtaminen sekä kellonajan tarkkana pitäminen. Työssä tarkastellaan myös suunnitellun laitteen lopputulosta, lisäominaisuuksien toimivuutta sekä vaihtoehtoisia menetelmiä lopputuloksen parantamiseksi. Työssä toteutettu laite ei onnistunut täysin suunnitellun mukaisesti. Suunnitellut elektroniset ja ohjelmistolliset ominaisuudet saatiin toteutettua lähes halutunlaisesti, mutta kellon heikon mekaniikan ansiosta komponenttien integrointi kelloon epäonnistui osittain.

Avainsanat: Raspberry Pi 2, valovastus, servomoottori, verkkosivu

Afflekt A.N. (2018) Bachelor's Thesis. University of Oulu, Degree Programme in Electronics and Communications Engineering, Bachelor's thesis, 38 p.

ABSTRACT

This bachelor's thesis is about connecting mechanical, analogous, battery operated alarm clock into local area network, making it a sort of IoT-device. Making device IoT capable also includes adding features such as: automatic time calibration and possibility to move clock pointers over LAN from the website. This thesis describes the design process and components required to make clock operational. Last part of this text consists of topics which discusses how the work was succeeded, also potential development targets and solutions are being talked about. Overall the work did not fully achieve the desired level as planned. Electronics and software parts were quite successful, but the integration of motors into surface of the clock was not successful due to weak mechanics of the clock.

Key words: Raspberry Pi 2, light dependent resistor, servo motor, web page

SISÄLLYSLUETTELO

ALKULAUSE

Kiitän Juha Häkkistä kattavasta ja avuliaasta ohjauksesta työni parissa. Erityisesti olen kiitollinen mahdollisuudestani saada hyvää etäohjausta, joka on mahdollistanut kandidaatintyöni viimeistelyn vaihto-opiskeluni aikana.

Prahassa 25.11.2018

Arttu Afflekt

LYHENTEIDEN JA MERKKIEN SELITYKSET

A/D	Analog/Digital
GPIO	General Purpose Input/Output
HTML	Hyper Text Markup Language
HTTP	Hyper Text Transfer Protocol
InGaN	Ingium gallium nitride
IoT	Internet of Things
LAN	Local Area Network
LED	Light Emitting Diode
NPN	(N-tyypin bipolaaritranstori)
PWM	Pulse Width Modulation
RTC	Real Time Clock

A	Ampeeri
E_v	Valaistusvoimakkuus
Φ_v	Valovirta
R	Resistanssi
U	Jännite
V	Voltti/Jännite

1. JOHDANTO

Asioiden internet on nykypäivänä paljon pinnalla oleva teknologia. Tämä on johtanut muun muassa siihen, että useita arkipäiväisiä laitteita on alettu yhdistämään verkkoon. Tällainen laite voi esimerkiksi olla kodinkone, joka välittää tilastaan tietoa verkkoon, tai kiuas, joka pystytään kytkemään päälle matkapuhelimella verkon yli, jotta sauna on optimaalisessa lämpötilassa saavuttaessa kotiin.

Tämän buumin innoittamana tässä työssä on tutustuttu IoT:n tuomiin mahdollisuuksiin sekä toimivuuteen, kun verkkoon liittyy vanha viisarikäyttöinen herätyskello. Työssä suunniteltiin kokonaisuus, joka kytkettynä Raspberry Pi 2 tietokoneeseen pystyi toimimaan tällaisen laitteen prototyyppinä. Aiheen valikoitumiseen vaikutti myös suuresti sen monipuolisuus.

Aluksi työssä esitellään laitteeseen potentiaalisia sensoriratkaisuja. Lisäksi kuvataan valitun sensorin valoportin suunnittelu laitteeseen. Tämän jälkeen tekstissä käydään läpi elektroniikan ja ohjelmiston suunnittelun vaiheet, jonka jälkeen toteutuneiden komponenttien toimintaa testattiin ja tuloksia tarkasteltiin. Lopuksi pohditaan kokonaisuuden onnistuneisuutta sekä esitetään joitakin ajatuksia kehityskohteista. Lisäksi mietitään, mitä olisi tullut tehdä toisin paremman lopputuloksen aikaansaamiseksi.

Työssä merkittävässä osassa on kellonajan kalibrointi. Työn loppupuolella pohditaan mahdollisia puutteita kalibroinnin tarkkuudessa. Tähän liittyen esitellään vaihtoehtoinen kalibrointimenetelmä, joka hyödyntää työssä käytettyä sensoriratkaisua, mutta tavoittelee tarkempaa lopputulosta.

2. SENSORIRATKAISUJA VIISARIN ASENNON TUNNISTAMISEEN

Markkinoilta löytyy kappaleen asennontunnistukseen sopivia sensoreita valtava määrä. Ainoastaan yhdessä komponenttien myyntiin erikoistuneessa verkkokaupassa on tarjolla sadoittain erilaisia vaihtoehtoja. Tässä kappalessa esitellään muutama potentiaalinen ratkaisu kellon viisarin asennon sensorointiin.

2.1. Kallistuskytkin

Kallistuskytkin on yksinkertainen sensori tilanteisiin, jossa halutaan tunnistaa kallistusta tai suuntausta. Aikoinaan kallistuskytkimissä käytettiin elohopeaa, mutta elohopean myrkyllisyyden vuoksi sen käytöstä on luovuttu. Kallistuskytkimen toimintaperiaatteena on tietyn kulma-arvon ylittyessä tai alittuessa kytkin vaihtaa tilaansa johtavaan tai ei johtavaan tilaan. [1]

Esimerkki nykyaikaisesta kaupallisesta kallistuskytkimestä on, C&K RB44145 kallistuskytkin, jossa elohopea on korvattu kullatulla pallolla. Kulman ylittäessä raja-arvonsa, kullattu pallo vierii sensorin sisällä kontaktien päälle, aiheuttaen muutoksen sensorin johtavuudessa. Edellä mainitun kallistuskytkimen liipaisukulma on n. 10° [2]

2.2. Magneettinen kulmasensori

Markkinoilla on tarjolla lukuisia magneettisuuteen perustuvia kulmasensoreita. Magnetismiin perustuvat puolijohdesensorit tarjoavat huomattavasti kallistuskytkintä tarkempaa informaatiota kappaleen asennosta. Esimerkiksi AKM EM3242 kulmasensori tarjoaa 360° kulman tunnistuksen 25 °C lämpötilassa +/- 3° virheellä. [3]

Useat magneettiset kulmasensorit perustuvat Hall-ilmiöön. Hall-ilmiöksi kutsutaan ilmiötä, jossa ulkoinen magneettikenttä aiheutta muutoksen sähkövirran kulkusuunnassa. Ilmiö syntyy, kun Lorentz-voima kohdistaa liikkeessä olevaan varattuun hiukkaseen voiman. Kun ulkoinen magneettikenttä tuodaan puolijohdelevylle, jossa kulkee virta, aiheuttaa magneettikenttä sähkövirran kulkusuunnassa muutoksen. Levyn välille syntyy ilmiön seurauksena mitattavissa oleva potentiaaliero, jota kutsutaan Hall-jännitteeksi. [4,5 s. 63 - 64]

2.3. Viisarin asennon tunnistaminen LDR/LED-yhdistelmällä

Työssä toteutetun laitteen mekaniikan vuoksi, sensoriratkaisuksi valittiin valovastukseen ja LED:n perustuva valoportin kaltainen menetelmä. Tässä kappaleessa kuvataan valoportin suunnittelun kannalta vaadittu teoria.

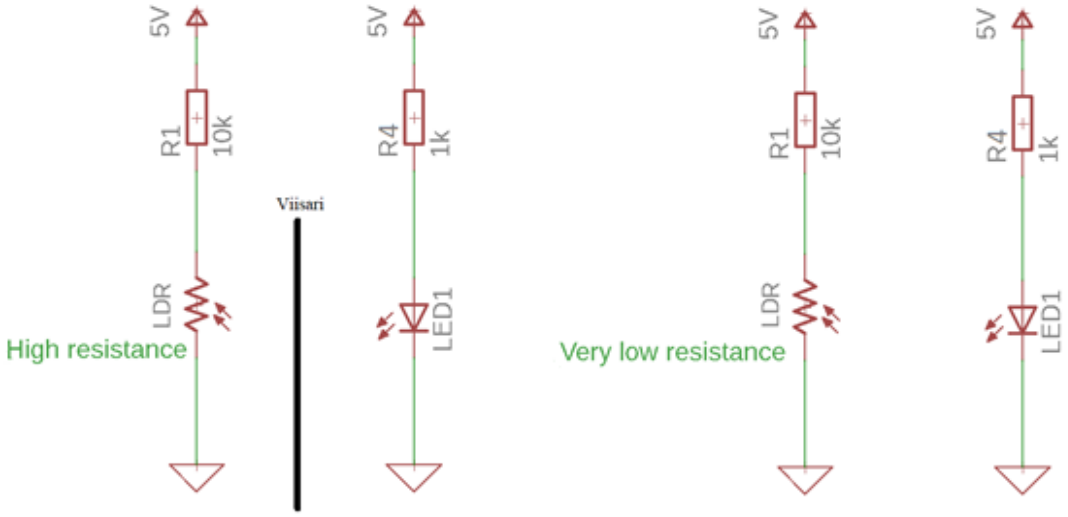
Valoportti muodostettiin asettamalla LED viisareiden yläpuolelle ja valovastuksen pinta asetettiin viisareiden alle kello 12 kohdalle. Valovastuksen yli näkyvä jännite tuotiin komparaattorille, jonka perusteella Raspberry Pi:lle välitettiin tieto, ovatko viisarit oikeassa kohdassa

Valovastuksena käytetyn VT90N2:n datalehdestä huomataan, että kyseessä on komponentti, jonka resistanssi riippuu sen pinnalle kohdistuvan valaistusvoimakkuuden määrästä. [6] Valaistusvoimakkuus kuvaa määritelmänsä mukaisesti pinta-alalle kohdistuvaa valovirran määrää. [7]

Toinen valoportin muodostavista komponenteista on LED. Valoportissa käytettiin valkoista valoa emittoivaa InGaN LED:iä. [8] Tavallisten LED:n valaistusvoimakkuus on noin 1 – 1000 mcd suuruusluokkaa, mutta tässä työssä käytetyn LED:n emittoiman valon intensiteetti on datalehden mukaan välillä 10 – 12 cd. [8,9].

LED:ksi valittiin siis erittäin kirkas LED. Koska LED:n valovoima on suuri, on myös sen lähettämä valovirta suuri. [7] Koska tässä työssä LED:n ja valovastuksen välinen etäisyys oli hyvin pieni, oletettiin että kaikki LED:n valovirta kohdistuu valovastuksen pinnalle.

Valovastuksen datalehdestä huomataan, että komponentin resistanssi on riippuvainen valaistusvoimakkuudesta. LED:n emittoiman valon intensiteetti oli suuri, ja sen emittoima valovirta kohdistui pääasiassa valovastuksen pinnalle. Valaistusvoimakkuuden määritelmän perusteella tehtiin johtopäätös, jonka mukaan LED:n valon kohdistaminen valovastuksen pinnalle, aiheuttaa erittäin suuren muutoksen valovastuksen resistanssissa. Valovastus sijoitettiin sarjaan vastuksen kanssa kuvan 1 mukaisesti. Tätä kytkentää testattiin erilaisissa huoneistoin valaistuksissa ja saatiin tuloksiksi seuraavanlaisia arvoja:



Kuva 1. Valovastuksen ja LED:n toimintaperiaate viisarin asennon tunnistukseen.

Tilanne, joka kuvastaa viisareiden osoittavan klo 12, eli viisarit estävät LED:n valon kohtisuoraa pääsyä valovastuksen pinnalle saatiin jännitteeksi valovastuksen yli.

$$U_{LDR} = 5\ V\ (pimeä\ huone)$$
$$U_{LDR} = 2{,}5\ V\ (kirkas\ huone)$$

Tilanne, jossa LED:n emittoima valo saavutti valovastuksen, saatiin seuraavanlaiset jännitteet. Havaittiin, että kasvattamalla LED:n läpi kulkevaa virtaa vastusta R4 pienentämällä, saatiin valovastuksen yli olevaa jännitettä hieman matalammaksi.

$$U_{LDR} = 0{,}16\ V\ (R4 = 3{,}2\ k\Omega)$$
$$U_{LDR} = 0{,}10\ V\ (R4 = 1\ k\Omega)$$

LED:n valon osuessa valovastuksen pinnalle huoneistoin valaistuksella ei havaittu olevan vaikutusta jännitteeseen.

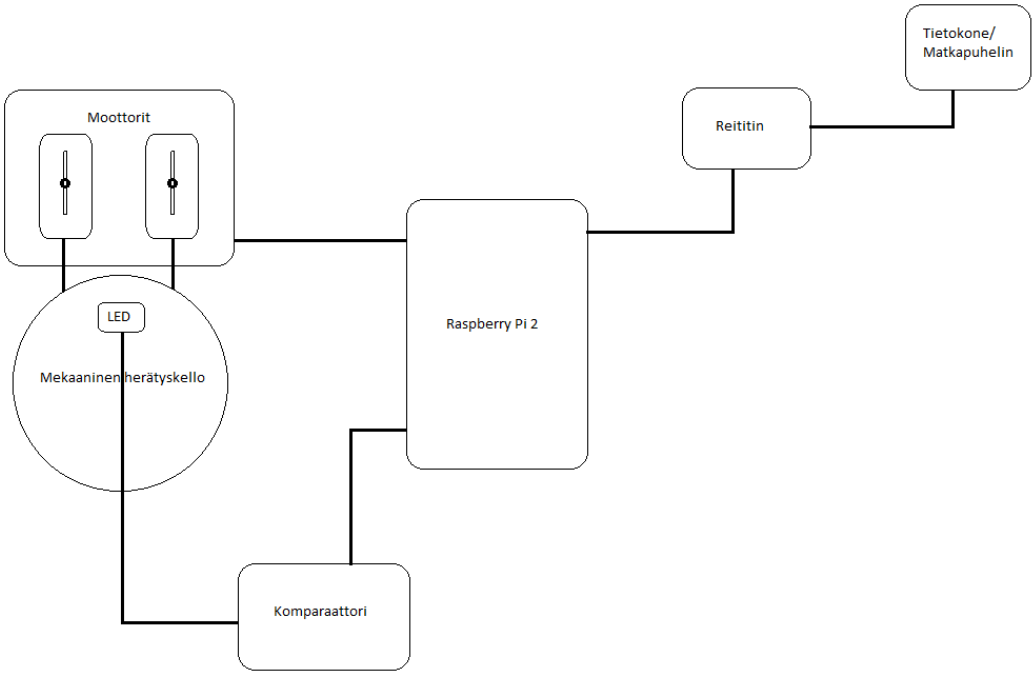
Tehdyn kokeen perusteella todistettiin aiemmin esitetty oletus siitä, että LED:n emittoima valo aiheuttaa suuren muutokset valovastuksen resistanssiin. Tämän lisäksi havaittiin, että valovastuksen resistanssi on huomattavasti voimakkaammin riippuvainen sen yläpuolella sijaitsevan LED:n emittoimasta valosta, kun ympäröivän tilan valaistuksesta.

3. KOMPONENTTIEN LIITTÄMINEN RASPBERRY PI:LLE

Laitteen funktionaaliseen toiminnallisuuteen osallisena olevat komponentit ovat esillä kuvassa 2 olevassa lohkokaaviossa. Herätyskello on itsenäinen koneisto, jonka käyntiin ei ollut tarkoitus vaikuttaa tässä työssä lisätyillä ominaisuuksilla. Raspberry Pi:n tarjoamien ominaisuuksien vuoksi kellon näyttämää aikaa sekä herätyksen ajankohtaa oli tarkoitus voida säätää lähiverkossa toimivan verkkosivun välityksellä. Kellon- ja herätysajan säätämisen tuli tapahtua moottoreiden avulla. LED:n ja komparaattoripiirin avulla laitteen kellonaika oli tarkoitus kalibroida päivittäin.

Järjestelmän toimintaa ohjasi Raspberry Pi 2 tietokone. Raspberry Pi tarjoaa rajapinnan kellon viisareita liikuttavien moottoreiden ohjaukselle ja kalibrointi-LED:n tilan lukemiselle. Tämän lisäksi Raspberry toimii palvelimena tarjoten verkkosivun kellon viisareiden ohjaukseen lähiverkossa.

Tietokone/matkapuhelin mahdollistaa yhteyden ottamisen samassa lähiverkossa olevan reitittimen kautta Raspberry Pi:llä olevaan kellon hallintaverkkosivuun.



Kuva 2. Järjestelmän lohkokaavio

3.1. Raspberry Pi 2

Raspberry Pi on yhdellä, noin luottokortin kokoisella piirilevyllä oleva tietokone, jota kehittää Raspberry Pi Foundation niminen yritys. [10] Raspberry Pi:stä on olemassa useita eri malleja, joista tässä työssä käytettiin mallia Raspberry Pi 2. Raspberry Pi 2 perustuu Broadcomin BCM2836 piirisarjaan, joka tekee siitä lähes identtisen samaan piirisarjaan perustuvan Raspberry Pi 1:n kanssa. Merkittävimpänä erona Raspberry 1:een on uusi prosessori. [11]

Raspberry Pi sisältää kaikki käyttäjälle näkyvät ominaisuudet missä perinteinenkin PC. Tämän lisäksi Raspberry:ssä on käytössä ohjelmoitavia GPIO-pinnejä, joiden avulla saadaan Raspberry kommunikoimaan monien muiden elektronisten

komponenttien kanssa. GPIO-pinnien lisäksi Raspberry:llä on tietokoneen tavoin tuki internetyhteydelle, joka mahdollisti kätevästi lähiverkon kautta tapahtuvan etäohjauksen.

3.2. Valoportin signaalin kytkeminen Raspberrylle

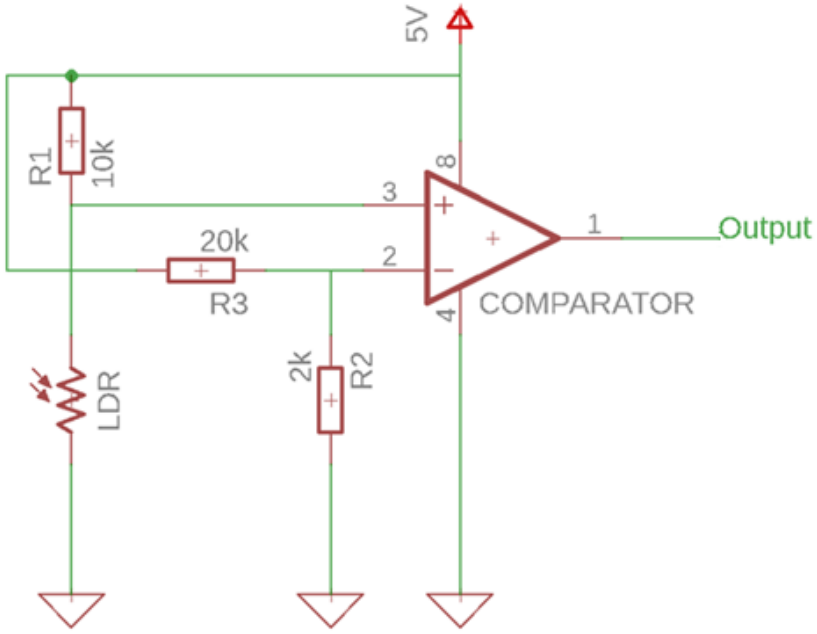
Raspberry Pi 2 tarjoaa lukuisia GPIO-pinnejä, luoden täten sähköisen rajapinnan liittää tietokoneeseen muita elektronisia komponentteja. Nämä GPIO-pinnit voidaan konfiguroida toimimaan tulo, lähtö tai esimerkiksi sarjaliikenneväylänä. [12]

Valoportin signaalia varten GPIO-pinni on asetettava toimimaan tulona. Kun GPIO-pinni konfiguroidaan toimimaan tulona, asettuu se kelluvaan tilaan. [13] Tästä johtuen tulona oleva GPIO-pinni voidaan joko vetää ylös 3,3 V jännitteeseen, tai alas maihin. Tämä voidaan tehdä joko vaikuttamalla Raspberryn Pull up tai Pull Dn Control rekistereihin tai vaihtoehtoisesti asettamalla fyysiset vastukset ajamaan saman asian. [13] Kuten aikaisemmin huomattiin, Raspberry Pi tarjoaa GPIO-pinnin kautta tuen vain digitaaliselle tulolle, täytyi seuraavaksi suunnitella luvussa 2 esitetyn valoportin jännittesignaalille 1-bittisenä A/D-muuntimena toimiva komparaattoripiiri.

Komparaattoripiirin suunnittelussa hyödynnettiin luvussa 2 mitattuja havaintoja ja arvoja siitä, kuinka huoneiston valaistus ja LED:n emittoiva valo aiheuttivat muutosta valovastuksen resistanssiin. Näiden tietojen perusteella komparaattorin referenssijännite asetettiin sopivalle tasolle jännitejaon kautta ja se tuotiin komparaattorin invertoivaan tuloon. Jännite mitoitettiin alla olevan jännitejaon mukaisesti.

$$V_{-} = U_{Ref} = \frac{R2}{R3+R2} * 5\,V = 454{,}45\,mV \qquad (1)$$

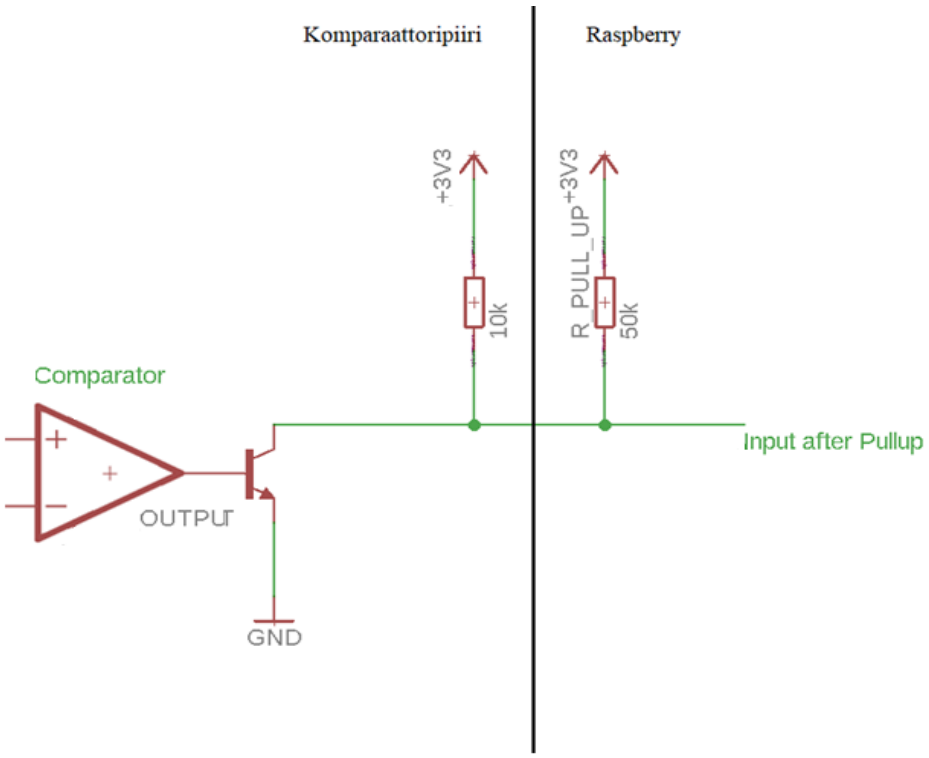
Referenssi jätettiin tarkoituksella huomattavasti mitattua 0,10 V suuremmaksi, koska signaalin luonteen vuoksi täten voitiin välttyä hystereesin asettamiselta piirille. Kuva 3 esittää mitoitettua komparaattorikytkentää.



Kuva 3. Mitoitettu komparaattorikytkentä.

Komparaattoripiiri kytkettiin Raspberryn GPIO22-pinniin. Tulona toimivien GPIO-pinnien 3,3 V jännitevaatimuksesta johtuen työhön valittiin komparaattoriksi Texas Instrumentsin LM339 Quad Differential Comparator. Kyseinen komparaattori on laajalla käyttöjännitealueella varustettu avokollektorilähtöinen komparaattori. Tilan vaihdos ilmenee siten, että lähtö voi joko kellua, tai transistori voi olla johtavassa tilassa. Komparaattorin ollessa johtavassa tilassa, lähtö on vedetty maahan. Kyseiseen komparaattoriin päädyttiin siksi, että lähdön virtanielu on riippumaton komparaattorin käyttöjännitteestä, joten se tekee sen mitoittamisen Raspberry Pi:n GPIO-pinniin helpoksi. [14]

Kuva 4 esittää komparaattorin ja GPIO-pinnin kytkentää. Kuvasta nähdään, että kytkentään on lisätty Raspberry sisäisen n. 50 kΩ vastuksen lisäksi ulkoinen 10 kΩ vastus. [15] Ulkoinen vastus lisää toiminnan luotettavuutta huomattavasti, koska sisäisiä vastuksia ohjaavat kontrollirekisterit, joiden resetoituessa vastukset lakkaavat toimimasta.



Kuva 4. Komparaattorin kytkeytyminen Raspberryyn.

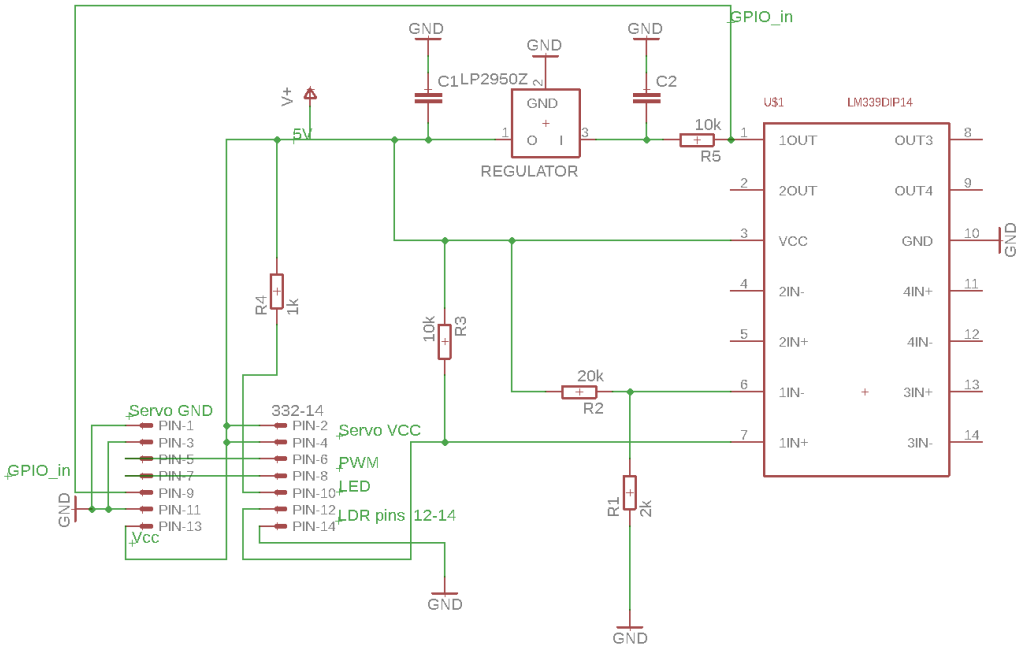
3.3. Moottoreiden kytkeminen Raspberrylle

Herätysaikaa muuttavaksi moottoriksi valittiin DFRobot DMS-MG90 180° servomoottori. Kyseessä on 180° liikeradalla oleva mikroservomoottori. Päätelmä servon sopivuudesta työhön tehtiin datalehdessä mainitun ”stall current” virran perusteella. Tämä on siis suurin mahdollinen virran määrä, jota moottori voi vetää (300 mA). Tällaiseen tilanteeseen ei kuitenkaan edes pitäisi käytössä joutua, joten oletettiin, että moottori on riittävän pienitehoinen. Servomoottorin kulman asentoa säädetään pulssinleveysmodulaatiolla. [16]

Kellon kesä- ja talviajasta sekä ajan kalibroinnista vastaa DFRobot Micro Servo 9g 360 (DF9GMS). Kyseessä on mikroservon kaltainen jatkuvasti pyörivä moottori. Moottorin suurimmasta mahdollisesta virrasta ei ole mainintaa datalehdessä. Ilmoitetun suurimman vääntömomentin perusteella kuitenkin oletettiin, että moottori on hieman toista moottoria heikkotehoisempi. Näiden tietojen puitteissa ja samankaltaisen helppokäyttöisyyden perusteella moottoria pidettiin oivana valintana työhön. Tätäkin moottoria ohjataan hyödyntäen pulssinleveysmodulaatiota, mutta tässä tapauksessa PWM säätää moottorin pyörimisnopeutta- ja suuntaa kulman sijaan. [17]

Moottoreiden kytkeminen Raspberrylle oli huomattavasti yksinkertaisempaa, kun kalibrointipiirin kytkeminen. Koska moottoreiksi valittiin pienivirtaiset servomoottorit, voitiin moottoreiden käyttöjännite kytkeä suoraan Raspberryn 5 V:n käyttöjännite pinnistä. Raspberryn GPIO-pinnit 2 ja 4 ovat kytketty suoraan sen virtalähteeseen, joten virtalähteen spesifikaatiot ja laatu ovat suuressa roolissa. Moottorit ja komparaattoripiirillä olevat komponentit saavat käyttöjännitteensä suoraan Raspberryn 5 V ja GND pinneistä, jotka sijaitsevat GPIO2,4,6,20-pinneissä. [18] Moottoreiden kolmas pinni kytkettiin suoraan lähdöiksi asetettuihin GPIO16- ja GPIO18-pinneihin. Näiden GPIO-pinnien funktio oli tuottaa PWM:ta joilla moottoreita voitiin ohjata. Servojen datalehdissä ei ollut mainintaa PWM johtimen virrasta. Koska kyseessä oli vain ohjaussignaali, oletettiin että sinne ei virtaa juurikaan kulkenut.

Kuvassa 5 on esitetty valmiin kytkennän piirikaavio. Kytkennässä on luotu liitynnät Raspberrylle sekä moottoreille, jotka yhdistyvät piirikaavion muihin komponentteihin.



Kuva 5. Koko piirilevyn piirikaavio

4. SÄHKÖISTEN KOMPONENTTIEN OHJAAMINEN VERKKOSIVUN VÄLITYKSELLÄ

4.1. Raspbian ja Apache

Raspberry Pi:n käyttöjärjestelmänä on UNIX-pohjainen Debian jakelupaketin pohjalta kehitetty Raspbian käyttöjärjestelmä. [19] Yhdessä Apache HTTP-palvelinohjelman kanssa, Raspbian tarjoaa alustan kellon moottoreita ohjaaville Python-koodeille, sekä mahdollisuuden ottaa käyttäjältä käskyjä vastaan verkkosivun välityksellä. [20] Jotta moottorien ohjaus verkkosivun kautta olisi mahdollista, täytyi Linuxista sallia oikeuksia eri käyttäjille. Erityisesti palvelinta operoiva käyttäjä ”www” tarvitsee kattavammat oikeudet muun muassa GPIO-pinnien käyttöön.

4.2. Verkkosivu ja Python

Kellon etähallintaa varten tehtiin yksinkertainen verkkosivu. Verkkosivun runko toteutettiin HTML-kielellä. Jotta interaktio käyttäjän kanssa olisi mahdollista, lisättiin verkkosivuun dynaamisia ominaisuuksia hyödyntäen PHP-skriptauskieltä.

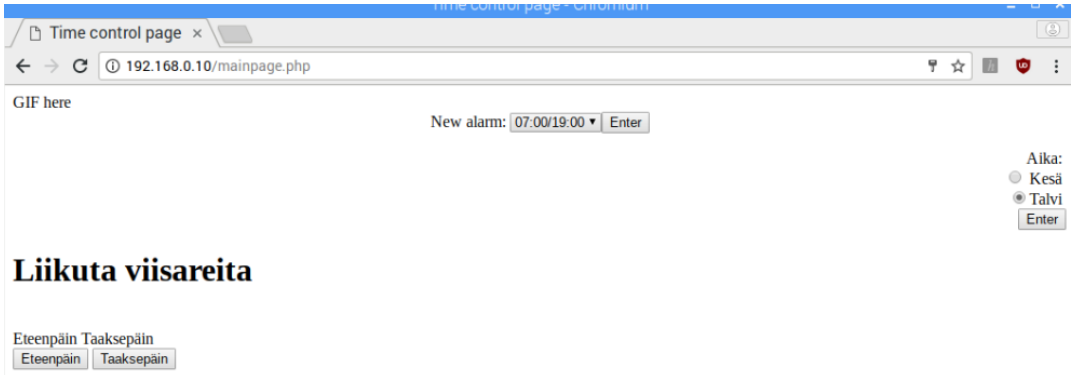
Käyttäjän muuttaessa kellon asetuksia verkkosivulla, hyödynnettiin HTML-kielen ominaisuutta lomakkeet (forms) käyttäjän tiedon keräämiseen. [21] Lomakkeen avulla saatiin tieto välitettyä PHP-skriptille hyödyntäen PHP:n metodia ”$_POST[]” [22]. PHP tarjosi tuen suorittaa komentoja hyödyntäen käyttäjän antamaa tietoa verkkosivulta. Tämä oli mahdollista käyttäen PHP:n shell_exec($suoritettava_komento) komentoa. [23] Tässä toteutuksessa komennon välittämisen toimivuutta parannettiin suorittamalla PHP:n komento escapeshellcmd() suoritettavalle komennolle. Edellä mainittu PHP:n komento poistaa tekstistä merkit, joilla voitaisiin yrittää huijata suoritettavaa komentoa. [24]

Todellinen moottorien ohjaus tapahtuu Python-skripteillä, jotka käynnistetään PHP:n toimesta. Skriptit saavat tietonsa kellon tilan muutoksista komentoriviargumenttinä PHP:n välittämänä. Kesä- ja talviajan tieto Pythonilla on tämän konfiguroimiseen tarkoitetussa tekstitiedostossa.

Näiden tietojen perusteella koodit hyödyntävät GPIO-pinnien käyttöön tarkoitettua RPi.GPIO-moduulia. Tämän moduulin avulla voitiin muun muassa alustaa GPIO-pinnit haluttuun tilaan, sekä ohjata moottorien nopeutta tai kulmaa PWM:llä. [25]

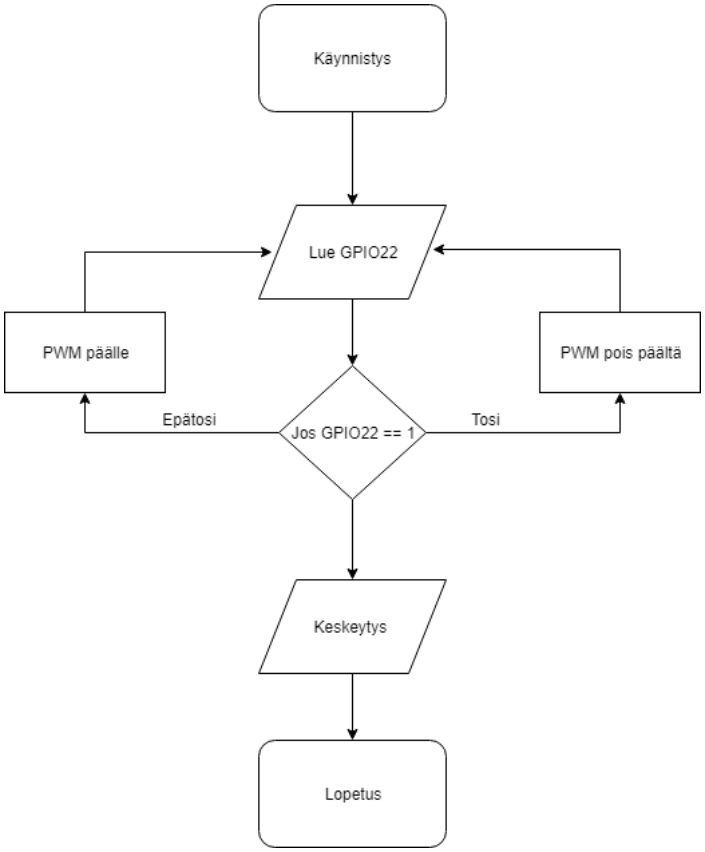
Kellonajan kalibroinnin käynnistämiseen olisi täytynyt hyödyntää Linuxin cron-komentoa, joka täytyi ajoittaa suorittamaan kellonajan kalibrointikoodi haluttuun aikaan klo 00:00. Kellonajan kalibrointia varten toteutettiin niin ikään RPi.GPIO-kirjastoa hyödyntävä Python koodi. Koodi lukee tuloksi konfiguroitua GPIO-pinniä, while-silmukassa. GPIO:lta luettu tila kertoo, täytyykö viisareita liikuttavaa moottoria käyttää. Tähän työhön ei cron-komentoa toteutettu, sillä oletettiin, että testaamalla ratkaisun toimivuutta eri valaistuksissa kertoisi enemmän ratkaisun toimivuudesta.

Kellon ohjauksessa käytetty verkkosivu on esillä kuvassa 6. Alasvetovalikosta New alarm: on mahdollista antaa moottorille käsky, joka muuttaa moottorin kulmaa uuden ajan vaatimalla tavalla. Sivu sisältää myös ominaisuudet liikuttaa kellonaikaa eteen- ja taaksepäin, sekä vaihtaa kesäaikaa ja talviaikaa. Kellon ohjaussivun HTML-koodi on liitteessä 1.



Kuva 6. Kellon ohjaussivu.

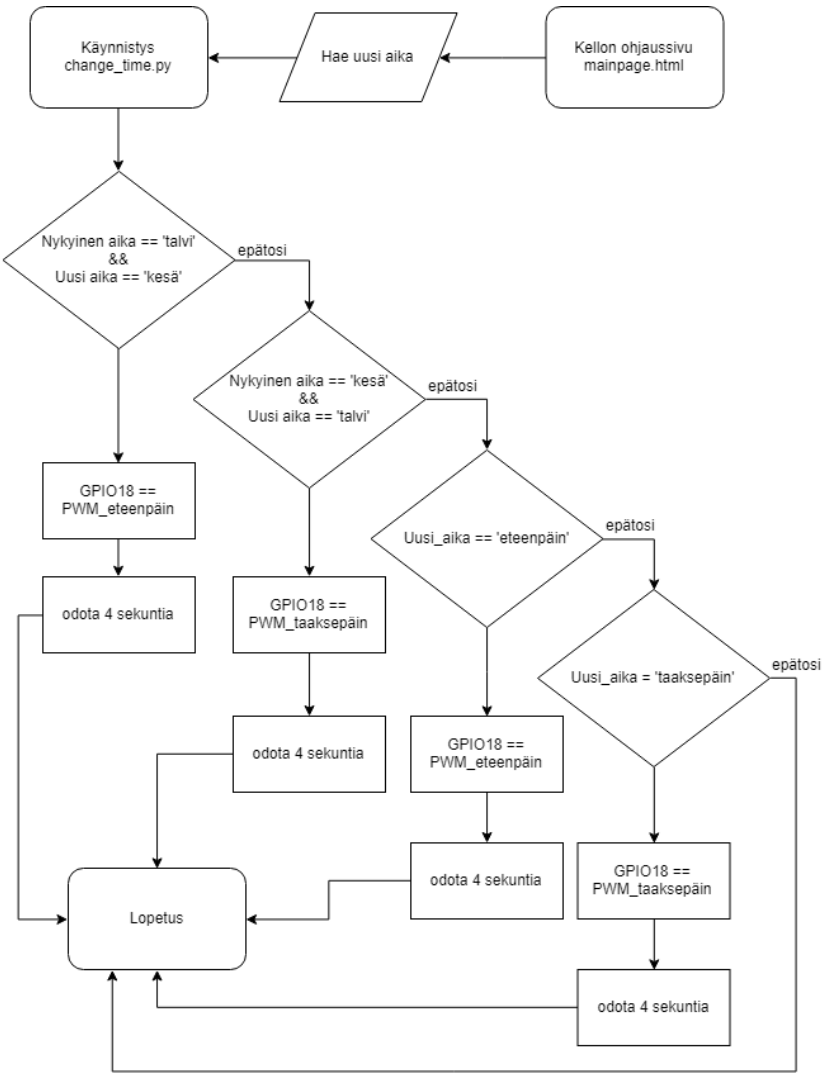
Kellonajan kalibroinnin suorittavan koodin vuokaavio on esitetty kuvassa 7. Koodi lukee tulona toimivaa GPIO22-pinniä silmukassa. Tulo-GPIO:n ollessa tilassa Epätosi, syöttää se PWM-signaalia viisaria kääntävälle moottorille, kunnes tulosignaali nousee takaisin tilaan Tosi. Kuvassa 7 oleva koodi on muokattu testikäyttöä varten. Suunnitellussa toiminnassa lohko PWM tulisi korvata silmukasta poistumisella ja GPIO-pinnien vapauttamisella. Ajan kalibrointi koodi on liitteessä 2.



Kuva 7. Ajan kalibrointi koodin vuokaavio.

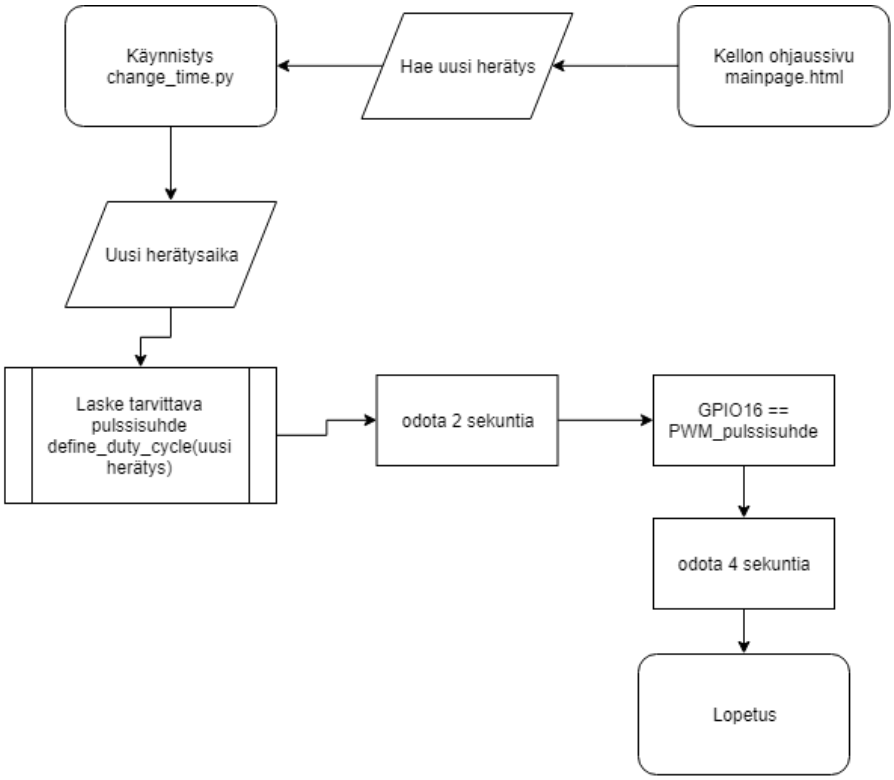
Kellon ohjauksen toiminnallisuutta havainnollistaa alla olevat kuvat 8 ja 9. Kuva 8 esittää koodia, joka hoitaa kellonajan muuttamista kesä- ja talviajan välillä. Uusi aika haetaan ohjaussivulta muuttujana, joka syötetään PHP-koodin kautta Pythonille kappaleen 4.2 mukaisesti. Vuokaaviossa esiintyvä muuttuja ”Nykyinen aika” luetaan koodissa tekstitiedostosta, jonne asetettu aika lopuksi tallennetaan. Odota-tilat vuokaaviossa ovat Pythonin funktiota time_sleep(), jolla ainoastaan varmistetaan GPIO-operaatiolle riittävästi aikaa suorittua. Koodissa oletettiin, että viisari pyörähtää tunnin verran neljässä sekunnissa.

Lohkokaaviossa esiintyvät operaatiot PWM_eteenpäin ja PWM_taaksepäin toteutettiin koodissa esiintyvien muuttujien dc_forwad_p ja dc_backward_p avulla. Koska moottorin pyörimisnopeutta sekä pyörimissuuntaa ohjattiin PWM:lla, määrättiin edellä mainituilla muuttujilla pulssisuhde, jonka oli tarkoitus toteuttaa haluttu suunta ja nopeus moottorille. Luomalla koodiin instanssi signal = GPIO.PWM(pinni, taajuus), pystyi luotu instanssi suorittamaan metodin start(pulssisuhde). Tällä metodilla saadaan syötettyä moottoria ohjaavalle GPIO-pinnille halutun taajuista oikean pulssisuhteen omaavaa kanttiaaltoa. Ajan muuttamisesta vastaavat koodit ovat liitteissä 5 – 6.



Kuva 8. Kesä- ja talviajan sekä viisarin muuttamisen koodien vuokaavio

Kuvassa 9 haetaan uusi herätysaika niin ikään kappaleen 4.2 metodien mukaisesti. Tämän tiedon perusteella lasketaan sopivan moottorin kulman aiheuttava pulssinleveys, jonka levyistä PWM-signaalia moottorille syötetään. Herätysaikaa ohjaavat koodit ovat liitteissä 3 – 4.



Kuva 9. Herätysajan asettamisen koodien vuokaavio.

5. TYÖN TULOKSET

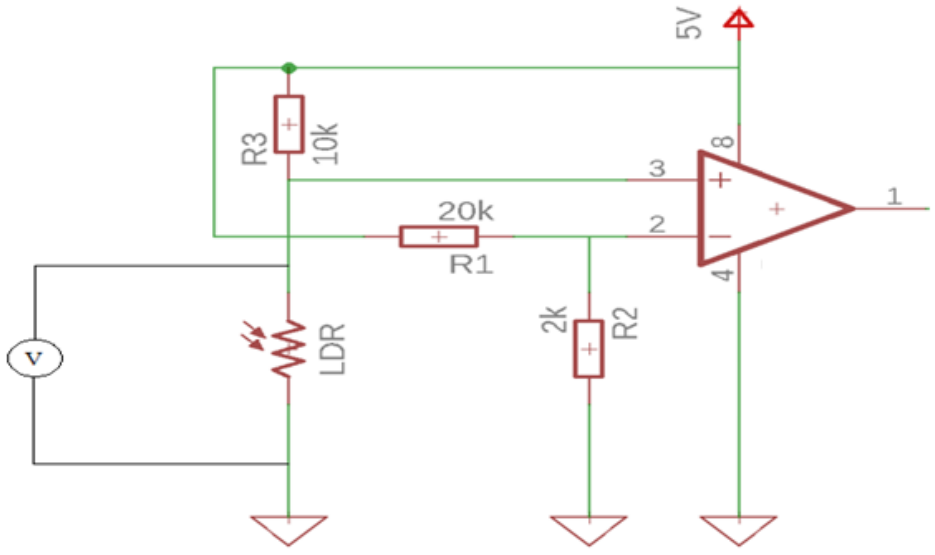
5.1. Yleisvaikutelma

Työn kokoamisvaiheessa ei onnistuttu aivan tavoitteiden mukaisesti. Kellon mekaniikka osoittautui yllättävän heppoiseksi, joten moottorien asentaminen ei ollut järkevää. Hammasrattaat, joilla välityksiä olisi muutettu olisivat kohdistaneet liikaa voimaa sivuttaissuunnassa säätönuppeihin. Moottorit saatiin kuitenkin liikkumaan verkkosivun kautta lähes suunnitellusti, joten elektroniikka ja ohjelmistot saatiin kuitenkin toimimaan alkuperäisen suunnitelman mukaisesti.

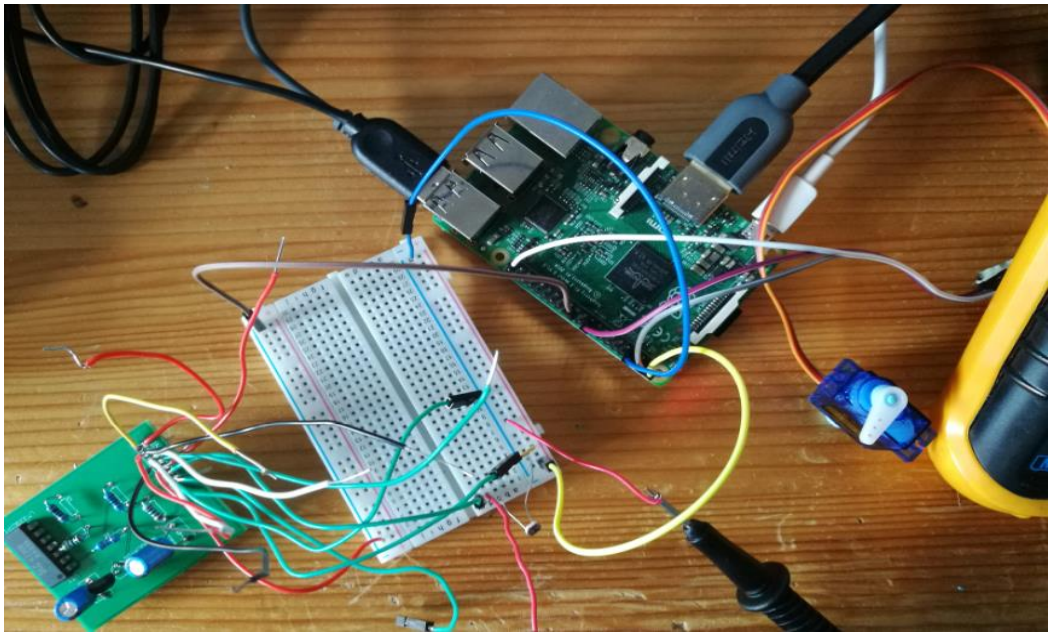
Koska kesä- ja talviajan sekä ajan kalibrointiin käytetyn moottorin toiminta käsitti huomattavamman määrän työhön suunniteltuja toiminta, päädyttiin testaamaan siihen liittyviä toimintoja. Vaikka työn lopullinen muoto jäi huomattavan prototyyppimäiseksi, laite kuitenkin pyrittiin kasaamaan sellaiseksi, että suunnitellut testit voitaisiin laitteelle suorittaa.

5.2. Kalibrointimenetelmän testaus

Ensimmäisessä laitteen mittauksista oli tarkoitus testata, saadaanko kellonaikaa kalibroiva moottori pyörimään ja pysähtymään muutamalla valovastukselle kohdistuvaa valaistusta. Onnistuneeseen tulokseen moottorin olisi pitänyt käydä ainoastaan, kun valovastuksen yli oleva jännite on lähellä spesifioitua n. 450 mV. Valovastuksen yli oleva jännite mitattiin kytkennästä kuvan 10 mukaisesti. Jännitemittarina käytettiin yleismittaria MASTECH MY64. Mittauskytkentä, joka mukaili kasatun laitteen toimintaa, on kuvassa 11.



Kuva 10. Jännitteen mittaaminen valovastuksesta.



Kuva 11. Mittauskytkentä.

Kytkennän lisäksi Raspberryllä käynnistettiin kalibrointia ohjaava Python-koodi ja saatiin mitattua taulukon 1 mukaiset tulokset. Tulosten perusteella huomattiin, että moottori toimi suunnitellusti. Kalibroinnin toteuttavan Python-koodin vuokaavio esiteltiin luvussa 4.2 kuvassa 7.

Taulukko 1. Moottorin toiminta ja valovastuksen jännite

Mittaus [nro]	V_LDR [V]	Moottorin tila [päällä/pois]	Tulos
1	4,31	Pois päältä	Ok. Moottori ei pyöri. Valaistus on vähäisä, joten jännite on järkevä
2	3,83	Pois päältä	Ok.
3	3	Pois päältä	Ok.
4	2,25	Pois päältä	Ok.
5	1,41	Pois päältä	Ok. Valaistus on jo kirkas
6	0,99	Pois päältä	Ok.
7	0,66	Pois päältä	Ok.
8	0,26	Päällä	Moottori pyöri kun jännite oli 0.26V. Sammui kun valo sammutettiin.

Taulukon 1 tulosten lisäksi, mitattiin kynnysjännitettä, jossa komparaattori vaihtaa tilaansa, eli moottorin tilassa tapahtuu muutos. Tämä testattiin siten, että riittävä valo kohdistettiin valovastukselle, jonka jälkeen valonlähdettä alettiin loitontamaan. Tämän seurauksena valovastuksen yli oleva jännite alkoi kasvamaan.

Moottorin pyöriminen pysähtyi, kun jännite oli noin 0,43 V. Jännite on hyvin lähellä suunniteltua 454 mV. Ottaen vielä huomioon, että kytkentä oli mitoitettu 5 V käyttöjännitteelle, ei virtalähteenä käytetty 5 V laturi kyennyt tuottaa kuin n. 4,83 V Raspberry Pi:n 5 V pinneille. Tämän perusteella jännitettä luettiin hyvinkin onnistuneesti.

Mittauksen aikana havaittiin, että moottori saattoi jumiutua tilanteissa, joissa kalibrointitapahtuma yritettiin toistaa uudelleen, ilman koodin käynnistämistä uudelleen. Tätä ei kuitenkaan pidetty kriittisenä ongelmana, koska suunnitellun toiminnan puitteissa koodi tulisi ajaa ainoastaan kerran, jonka jälkeen se sammutettaisiin.

5.3. Valovastuksen herkkyyden mittaus

Koska valovastuksen integrointi kellotauluun onnistui kohtuullisen hyvin, pystyttiin sitä testaamaan lähes alkuperäisen suunnitelman mukaisesti. Tarkoituksena oli mitata kuinka hyvin viisarit toimivat esteenä valolähteen ja valovastuksen välissä. Onnistuneena mittaustuloksena pidettiin tulosta, jossa viisarit pystyivät estämään valon pääsyn vastuksen pinnalle niin hyvin, että vastuksen resistanssi pysyi riittävän korkeana.

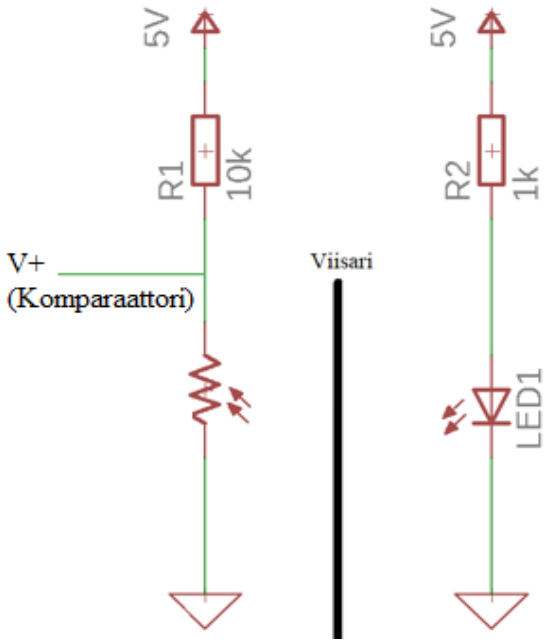
Kuten luvussa 3 esitettiin, komparaattorin referenssijännite oli asetettu noin 450 mV. Tämä tarkoittaisisitä, että kuvan 12. näyttämässä tilanteessa valovastuksen jännitteen pitäisi pysyä vähintään

$$V_+ > 450\,mV \tag{2}$$

Toisin sanoen vastuksen resistanssin tulisi olla suurempi kuin 989 Ω.

$$V_+ = 5\text{ V} * \frac{R_{LDR}}{R1+R_{LDR}} \tag{3}$$

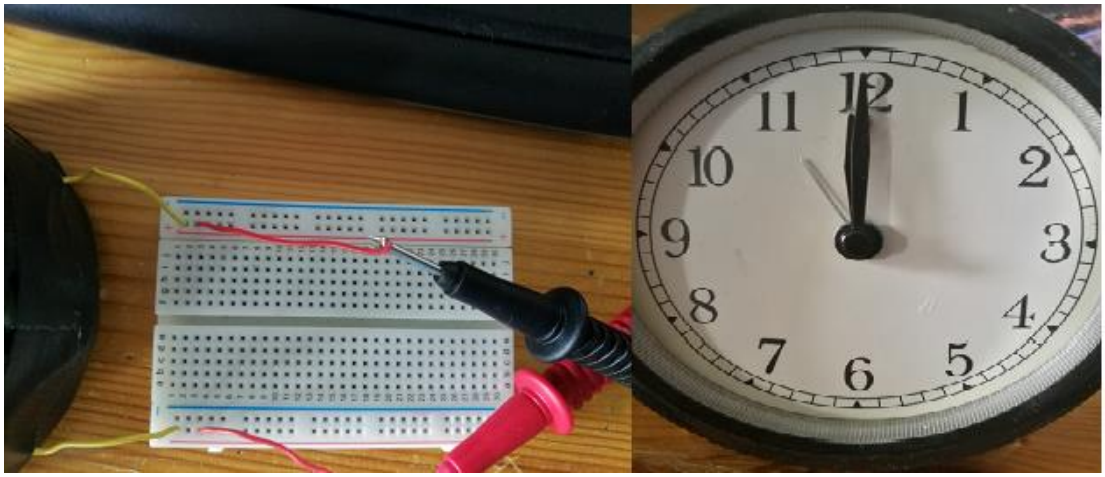
$$\frac{V_+ * R1}{5V-V_+} > R_{LDR} \rightarrow \frac{0.45V*10k\Omega}{5V-0.45V} > 989\ \Omega \tag{4}$$



Kuva 12. Valovastuksen toimintaperiaate, kun viisari on välissä.

Resistanssin mittauksessa käytettiin samaa yleismittaria, kun ensimmäisessä mittauksessa. Tämän lisäksi valaistusvoimakkuutta arvioitiin Huawei Honor 8:n valosensorilla Rohm BH1750. Valaistusvoimakkuus mitattiin asettamalla sensori noin kellon viisareiden yläpuolelle. Kellon viisarit asetettiin kuvan 15 mukaiseen asentoon, jonka jälkeen huoneiston valaistusvoimakkuutta säädettiin ja tulokset kirjattiin ylös. Mittauksista saatiin taulukon 2 mukaiset tulokset. Vastukselle menevät johtimet kytkettiin resistanssimittaukselle asetetulle yleismittarille koekytkentälevyn kautta.

Kuten kuvasta 13 huomataan, peittävät viisarit valovastuksen kauttaaltaan, joten odotettu tulos olisi se, että huoneistoin tuottamalla valaistuksella ei pitäisi olla häiritsevää merkitystä valovastuksen resistanssiin.

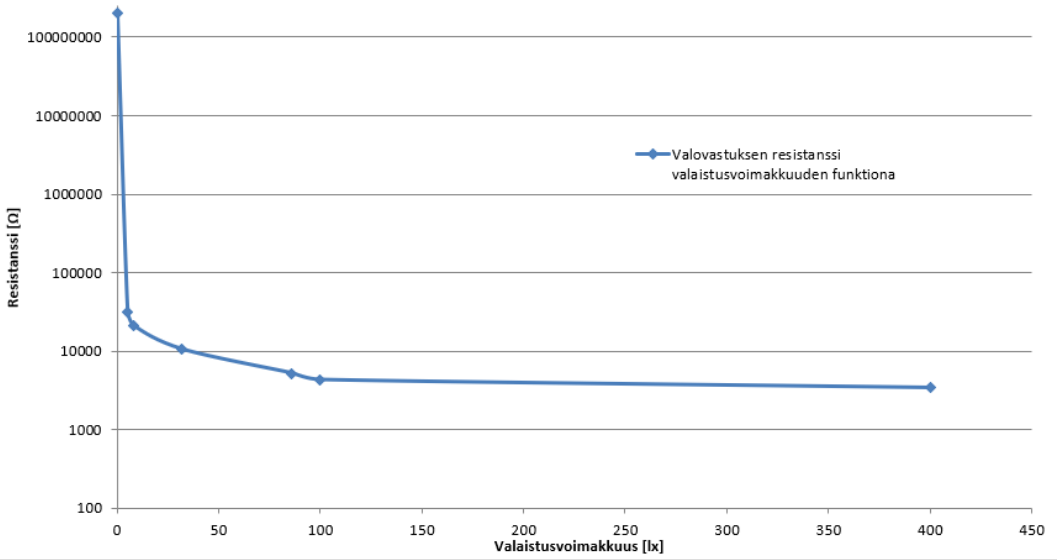


Kuva 13. Viisareiden asento mittauksen aikana.

Mittaukset 1 - 7 pyrkivät kuvastamaan tilannetta, jossa huoneiston valaistus on erilaisilla mahdollisilla tasoilla. Mittauksen 8 tarkoituksena oli kuvastaa tilannetta, jossa kello on ajallaan, mutta kalibrointi-LED on kohdistettu suoraan valovastuksen yläpuolelta. Kuvan 14 ja taulukon 2 perusteella voidaan olettaa, että sensoriratkaisu toimii suunnitellun mukaisesti, eli viisarit pystyvät estämään valon pääsyn sensorille, kun ne osoittavat kello 12.

Taulukko 2. Valovastuksen resistanssi valaistusvoimakkuuden mukaan.

Mittaus	**Resistanssi [Ω]**	**Valaistusvoimakkuus [lx]**
1	>200 MΩ	n. 0 lx
2	32 kΩ	5 lx
3	21 kΩ	8 lx
4	10.5 kΩ	32 lx
5	5.2 kΩ	86 lx
6	4.3 kΩ	100 lx
7	3.4 kΩ	400 lx
8	1.6 kΩ	n. 6000 lx



Kuva 14. Mitattu resistanssi valaistusvoimakkuuden funktiona.

Mittauksen aikana havaittiin, että vastuksen resistanssiin vaikuttaa myöskin sivussa päin olevat valonlähteet. Aikaisemmissa mittauksissa valaistusvoimakkuus mitattiin ainoastaan viisareiden yläpuolelta. Tästä johtuen suoritettiin myös tällaiselle tilanteelle mittaus, jossa menetelmän toimintaa pyrittiin häiritsemään kohdistamalla valoa sivusuunnasta.

Edellä mainitun kaltaisessa pahimmassa mahdollisessa tilanteessa vastuksen resistanssiksi saatiin ainoastaan noin 1,3 kΩ. Tämän tiedon perusteella kellon mekaniikkaa pystyttiin hyödyntämään suunnitellun kaltaiseen sensoriratkaisuun.

6. POHDINTA

6.1. Lopputulos

Tämän kandidaatintyön aiheeksi valikoitui kyseinen aihe siksi, koska aihe kuulosti alkuun monipuoliselta ja opettavaiselta. Työn edetessä aihe osoittautui monipuoliseksi niiltä osin, kun sitä osattiin odottaa. Tämän valossa kirjoittaja halusi oppia laajasta skaalasta asioita, ja sitä tämä työ tarjosi.

Laitteen lopputuloksesta ei kuitenkaan tullut täydellistä. Erityisesti haasteita tuotti mekaanisten ongelmien ratkaisu. Nämä haasteet olisi olleet korjattavissa, jos laite olisi tehty aivan alusta alkaen, eikä pyritty integroimaan jo olemassa olevaan kelloon lisäominaisuuksia. Tämä toteutustapa olisi lisäksi mahdollistanut huomattavasti tarkemman, tehokkaamman ja vapaamman tavan vaikuttaa laitteen toimintaan. Myöskin toteutettujen koodien toiminta oli virheherkkää. Koodien suorituksissa saattoi esiintyä odottamattomia virheitä. Mekaanisen toteutuksen epäonnistumisen vuoksi moottoreiden nopeuksia ja pyörimisaikoja ei koskaan mitoitettu realistisiksi, vaan ne jäivät prototyyppiasteelle. Yleisesti ottaen voidaan kuitenkin sanoa, että elektroniikan ja ohjelmistojen suunnittelussa onnistuttiin tarkoituksenmukaisesti, mutta fyysinen toteutus jäi ontumaan.

Kaikesta huolimatta, työssä kuitenkin onnistuttiin useassa asiassa. Kelloa ohjaavat moottorit saatiin pyörimään verkkosivun kautta, sekä komparaattoripiiri saatiin lukemaan valovastuksen jännitettä ja reagoimaan siihen oikein. Tämän perusteella voitiin todeta, että elektroniikan ja ohjelmistojen osalta funktionaalinen toiminnallisuus saavutettiin. Kokonaisuuden toimivuus jäi ontumaan juurikin mekaniikan puolesta. Mekaanisessa toteutuksen osalta, kalibroinnissa käytetty sensorimenetelmä kuitenkin toimi toivotunlaisesti, joten loppupeleissä ainoastaan moottorien integrointi kelloon epäonnistui täysin.

6.2. Yleiset kehityskohteet

Vaikka laitteessa funktionaalinen toiminnallisuus joiltakin osin saavutettiin, jäi työssä siltäkin osin huomattavasti kehitettävää. Vaikka kaikki suunnitellut ominaisuudet olisivat toteutuneet suunnitelman mukaisesti, olisivat ratkaisut olleet epäluotettavia ja epätarkkoja. Tässä kappaleessa käsitellyt asiat olisivat oletettavasti tehneet laitteen toiminnasta luotettavampaa, nopeampaa ja hyödyllisempää.

Eräs kehityskohde oli moottoreita ohjaava pulssinleveysmodulaatio. Tässä työssä pulssinleveysmodulaationa käytettiin Python kirjaston RPi.GPIO:n tarjoamaa pulssinleveysmodulaatiota. Tässä oli ongelmana se, että se oli ohjelmistotasolla luotua signaalia. Ohjelmistotason pulssinleveysmodulaation heikkoutena on muun muassa se, että laitteen suorittaessa muita prosesseja tulee signaaliin mahdollisesti viivästyksiä ja häiriöitä. Parempi vaihtoehto olisi sen sijaan laitteistotason pulssinleveysmodulaatio. Tämä tarjoaisi huomattavasti tarkemman ja nopeamman vaihtoehdon moottoreiden ohjaukseen, koska se ei olisi riippuvainen niin monesta muusta käyttöjärjestelmän tekijästä. Lisäksi moottoreita ohjanneiden Python-koodien toiminta saattoi olla epäluotettavaa ja niiden toiminnassa saattoi esiintyä häiriöitä. Tästä johtuen koodeissakin olisi varaa optimoinneille.

Kalibroinnissa käytetty sensorointimenetelmä ei ollut kaikista hienostunein vaihtoehto, vaikkakin se toimi yllättävän hyvin. Tämän luotettavuutta voisi kuitenkin mahdollisesti parantaa lisäämällä valovastuksen pinnalle optiikkaa, joka sallisi ainoastaan kohtisuorat valonsäteet. Tämä voisi vähentää ulkoisten valonlähteiden häiriötä. Parhaimmassa tapauksessa jopa LED:stä voitaisiin luopua kokonaan. Viisarin ollessa oikeassa kohdassa, ei linssi pystyisi suuntaamaan lainkaan valoa vastukselle. Tämän ongelmana olisi kuitenkin täysin pimeässä huoneessa tapahtuva kalibrointi, jolloin valoa ei olisi ollenkaan tarjolla. Toisin sanoen, vaikka silmiin pistävästä LED:stä luovuttaisiin em. tavalla, ei menetelmä siltikään olisi järin luotettava.

Tässäkin tapauksessa kellon alusta asti valmistaminen olisi ollut paras ratkaisu. Kellon moottori voitaisiin toteuttaa alusta asti vaikkapa askelmoottoreista ja kytkeä toteutukseen kulmasensorit, jotka välittäisivät tietoa viisareiden tarkasta kulmasta. Tämän avulla ulospäin näkyvistä ylimääräisistä osista päästäisin eroon, kalibrointi tarkentuisi ja sensorin tieto mahdollistaisi kalibroinnin useampiin kuin yhteen kellonaikaan.

Ohjelmistopuolella kehityskohteita on erityisesti verkkosivussa ja etäohjauksessa. Laitteen rakentaminen alusta asti mahdollistaisi luonnollisesti verkkosivulle huomattavan määrän lisäominaisuuksia. Lisäksi moottoreiden liikkeessä saattoi esiintyä bugeja. Erityisesti herätysaikaa säätävän moottorin käynti oli jokseenkin töksähtelevää. Vaikka ohjelmistojen puitteessa suunniteltu funktionaalisuus saavutettiin, oli ne huomattavan karkeita ja hidastoimisia.

Puhtaasti verkon yli tapahtuvaan ohjaukseen kehitettävää olisi tuelle ohjata ja lukea kellon tietoja internetin yli. Tämän ominaisuuden toteuttaminen vaatisi laitteelta huomattavasti enemmän tietoturvaa. Tässä työssä toteutettiin verkkosivu lähes täysin funktionaalisessa mielessä, sekä Linuxin käyttäjäoikeuksia jaettiin sitä mukaa, kun niitä tarvittiin.

Tietoturvallisen verkkosivun ja palvelimen toteuttaminen mahdollistaisi kellon etäkäytön lähiverkon ulkopuolelta. Käytännöllisestä näkökulmasta tämä ei kuitenkaan tarjoa mielestäni tärkeää lisää kellolle, sillä harva ihminen tuskin kokee tarpeelliseksi vaikuttaa herätyskelloonsa kodin ulkopuolelta käsin.

Lisäksi mainittakoon, että Raspberry Pi:n kaltainen tietokone on ylimitoitettu resurssi suuren virran kulutuksensa ja valtavan toiminnallisen potentiaalin kannalta. Tällainen ei ole kovin kestävä ratkaisu vanhan herätyskellon herättämiseen uudelle vuosituhannelle. Raspberry kuitenkin toimi tässä työssä ainoastaan erään prototyypin alustana.

Luonnollista olisi toteuttaa mahdollinen varsinainen tuote sitä varten kehitetyllä piirillä, jossa Raspberryn sijaan toimintaa ohjaisi mikrokontrolleri. Tämän ympärille voitaisiin sitten suunnitella vain välttämättömät komponentit. Täysin alusta asti toteuttavaan ratkaisuun kuuluisi myös muun muassa mekaniikan ja kellon rungon suunnittelu moottorin ympärille, joka olisi suoraan kytköksissä ohjauspiireihin. Tämän voisi ratkaista esimerkiksi askelmoottorilla, joka toteuttaisi kellon käynnin sekä ajan kalibroinnin. Näin päästäisiin eroon epätarkoista mikroservoista ja lopullinen toteutus olisi huomattavasti kompaktimpi ja ennen kaikkea tarkempi.

Kaiken kaikkiaan IoT-kello on mielenkiintoinen ajatus, siinä missä muutkin IoT-laitteet, mutta viisareilla toimivaan kelloon sitä ei ehkä ole järkevä kaupallisessa mielessä toteuttaa. Esimerkkinä mainittakoon kellon näyttämän ajan kalibrointi. Kello vaatii luonnollisesti jonkin toisen kellon esimerkiksi RTC:n, johon se vertaa aikaansa. Olisi luonnollisesti järkevää näyttää suoraan tarkan RTC:n perusteella aika

digitaalisesti, eikä käyttää tarjolla olevaa tarkkaa aikaa vain epätarkemman ajan kalibrointiin.

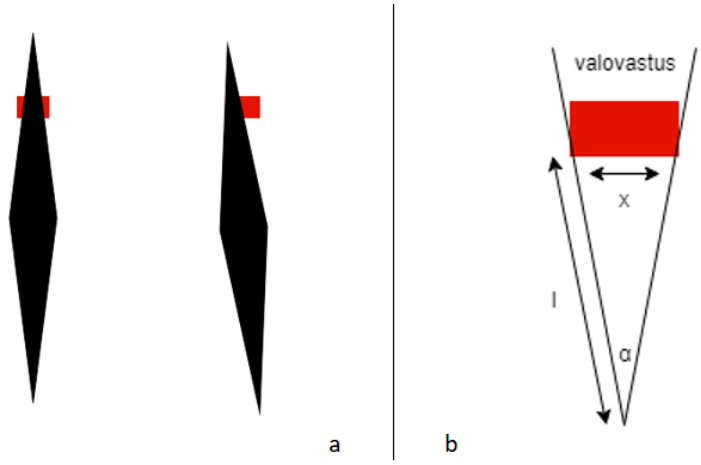
6.3. Kalibrointialgoritmin tarkkuuden parantaminen

Viisareiden asennon kalibroinnista muodostui työssä olennaisin sekä mielenkiintoisin ominaisuus. Kuten luvussa 4. esitettiin, viisareiden asento kalibroidaan ainoastaan yksinkertaisessa silmukassa, jossa tarkastellaan ainoastaan peittääkö viisari valovastuksen pinnan vai ei. Kuva 15a pyrkii havainnollistamaan kyseisen metodin aiheuttamaa epätarkkuutta viisareiden kalibroinnissa. Kuvassa 15a vasemmanpuoleinen kuvio esittää ideaalia tilannetta, jossa viisari osoittaa kohtisuoraa ylöspäin. Tässä tilanteessa valovastus on riittävän hyvin peitetty ja viisari tunnistetaan olevan oikeassa asennossa. Oikeanpuoleinen kuva esittää tilannetta, jossa viisari peittää valovastusta riittävästi, mutta ei ole kalibroitunut täysin oikeaan aikaan. Tämä on myöskin todennäköisempi asento viisarille pysähtyä, koska viisarin tulisi pysähtyä välittömästi komparaattorin tilan vaihtuessa. Tästä syystä johtuen tarkempi kalibrointialgoritmi voisi johtaa tarkkuuden parantamiseen.

Kuten kuvasta 15a huomattiin, valovastukselta saadaan todennäköisesti tieto viisarin oikeasta asennosta jo silloin, kun se ei vielä ole täysin oikeassa asennossa. Seuraavaksi esitellyssä kehitysideassa oletetaan yksinkertaisuuden vuoksi, kun viisarin reuna kohtaa valovastuksen välittyy tieto tilan vaihdoksesta eteenpäin. Todellisuudessa pelkkä reuna ei sitä vielä aiheuttaisi, vaan viisarin täytyisi peittää valovastusta hieman enemmän. Periaate ei sen sijaan muutu miksikään, mutta mittaustarkkuuden puitteissa on yhtä relevanttia tarkastella tilannetta hieman yksinkertaistetusti.

Kuva 15b havainnollistaa tilannetta, josta saadaan ratkaistua kulma α, jonka verran viisari liikkuu. Valovastuksen leveys d = 4,88 mm, viisarin pituus valovastukselle asti väliin asti l = n. 2,5 cm. Kuvassa 18. x = d = 4,88 mm ja l = viisarin pituus komponentin kulmaan = n. 2,5 cm. Kulma α saadaan laskettua kaavasta (5) ja sen arvoksi saatiin n. 11,2°.

$$\arcsin\left(\frac{x}{2l}\right) = \frac{\alpha}{2} \approx 5{,}6°,\ \alpha \approx 11{,}2° \tag{5}$$



Kuva 15a Alkuperäisen kalibrointimenetelmän epätarkkuus
Kuva 15b. Vaihtoehtoisen kalibrointimenetelmän parametrien mitoitus

Näiden tietojen perusteella kalibrointialgoritmi voidaan toteuttaa siten, että komparaattorin siirtyessä tilaan, joka edustaa kulman α määräämää sektoria, käynnistetään laskuri. Laskuri pysäytetään, kun komparaattorilta havaitaan, että sektori on ylitetty. Jos oletetaan, että kalibroinnin suorittava moottori on samalla akselilla kuin viisari, riittää tässä tilanteessa ainoastaan kiertää moottoria α / 2 – asteen verran taaksepäin.

Teoriassa yllä mainitun kaltainen kalibrointimenetelmä johtaisi tarkempaan lopputulokseen, kun nykyisellään käytetty. Yllä esitetyt arvot ovat todelliselle toteutukselle hieman liian suuret, koska tilan oletettiin vaihtuvan heti viisarin ja sensorin kohdatessa. Menetelmän implementointi vaatisi myös tarkempaa ohjausta moottoreille, tietoa kellon koneiston välityksistä sekä erittäin tarkkoja mittauksia kalibroinnin mitoittamiseen.

Esitelty menetelmä mahdollistaisi kuitenkin LDR/LED-yhdistelmän tarkemman hyödyntämisen viisarin asennon kalibroinnissa. On kuitenkin todennäköistä, että menetelmän hyödyntäminen vaatisi tehokkaampia ohjauskoodeja ja tarkempia moottoreita, kun tässä työssä on käytetty. Tämän takia pelkän uuden kalibrointialgoritmin toteuttaminen ei välttämättä olisi itsestään tehnyt työstä kokonaisuutena onnistuneempaa. Sen sijaan se olisi hyvin suurella todennäköisyydellä ollut huomattavasti tarkempi tapa hyödyntää työssä käytettyä sensorimenetelmää viisarin asennontunnistukseen.

7. YHTEENVETO

Työssä suunniteltiin valovastukseen ja LED:iin perustuva viisarin asennon tunnistusmenetelmä, joka muutettiin digitaaliseksi suunnitellun komparaattoripiirin kautta. Signaali kytkettiin Raspberry Pi:lle, joka ohjasi muun muassa tämän tiedon perusteella moottoreita, joidenka olisi alkuperäisen suunnitelman mukaan pitänyt pyörittää kellon moottoreita.

Raspberry Pi toimi myös lähiverkossa pyörivänä palvelimena, jossa oli verkkosivu, jonka kautta moottoreita pystyi pyörittämään. Työ oli monipuolinen ja opettavainen katsaus siinä käytettyihin komponentteihin. Valmis toteutus ei saavuttanut täysin alkuperäisen suunnitelman tavoitetta. Työn loppupuolella esiteltiin epäonnistuneet sekä puutteelliset toteutukset ja niihin pohdittiin vaihtoehtoisia ratkaisuja.

8. LÄHTEET

[1] adafruit by lady ada (luettu 15.10.2018) Tilt Sensor, The simple way to detect orientation or inclination. URL: https://learn.adafruit.com/tilt-sensor/

[2] C&K (luettu 15.10.2018) RB Series Rolling Ball Sensor Switch URL: https://www.ckswitches.com/media/1317/rb.pdf

[3] AsahiKasei AKM (luettu 15.10.2018) EM3242 Angle Sensor IC URL: https://media.digikey.com/pdf/Data%20Sheets/AKM%20Semiconductor%20Inc.%20PDFs/EM3242.pdf

[4] FYSA220/1 (FYS222/1) HALLIN ILMIÖ s. 1 - 2 (luettu 8.11.2018) URL: https://www.jyu.fi/science/fi/fysiikka/opiskelu/tyoosasto/tyot/fysa220/fysa220_1_k2016.pdf

[5] Jani Tuorila, Fysiikan laitos, Oulun yliopisto 4. huhtikuuta 2012 (luettu 8.11.2018) Kondensoidun materian fysiikka 763628S URL: https://wiki.oulu.fi/download/attachments/15698835/kmf.pdf?version=54&modificationDate=1334686671000&api=v2

[6] EXCELITAS TECH (luettu 18.9.2018) VT90N2 Datalehti URL: http://www.farnell.com/datasheets/612931.pdf

[7] Aaltoliike ja optiikka luentomoniste Seppo Alanko Kevät 2015 s. 96

[8] SHENZHEN CHENGGUANGXING INDUSTRIAL DEVELOPMENT CO.,LTD, Valkoisen valon LED 504WC. (luettu 18.9.2018) URL: https://www.arduino.cc/documents/datasheets/LED(white).pdf

[9] Silvonen Kimmo (2009) Elektroniikka ja puolijohdekomponentit. Gaudeamus Helsinki University Press / Otatieto, s. 126

[10] Raspberry Pi Foundation. (luettu 19.3.2018) What is Raspberry Pi? URL: https://www.raspberrypi.org/help/what-%20is-a-raspberry-pi/

[11] Raspberry Pi Foundation (luettu 19.3.2018) BCM2836 URL: https://www.raspberrypi.org/documentation/hardware/raspberrypi/bcm2836/README.md

[12] Broadcom Corporation (luettu 18.9.2018) BCM2835 ARM Peripherals (sivu 102-103) URL: https://www.raspberrypi.org/documentation/hardware/raspberrypi/bcm2835/BCM2835-ARM-Peripherals.pdf

[13] Ben Croston (luettu 19.3.2018) raspberry-gpio-python Inputs URL: https://sourceforge.net/p/raspberry-gpio-python/wiki/Inputs/

[14] Texas Instruments (luettu 5.8.2018) LMx39x, LM2901xx Quad Differential Comparators URL: http://www.ti.com/lit/ds/symlink/lm2901.pdf

[15] Raspberry Pi Foundation (luettu 5.8.2018) GPIO URL: https://www.raspberrypi.org/documentation/hardware/raspberrypi/gpio/README.md

[16] DF ROBOT (luettu 5.8.2018) 9g Metal Gear Micro Servo (1.8Kg). 180-asteen servon datalehti URL: https://media.digikey.com/pdf/Data%20Sheets/DFRobot%20PDFs/SER0039_Web.pdf

[17] DF ROBOT (luettu 5.8.2018) TowerPro SG90C 360 Degree Micro Servo Datalehti URL: https://media.digikey.com/pdf/Data%20Sheets/DFRobot%20PDFs/SER0043_Web.pdf

[18] James Adams, Raspberry Pi 2014 (luettu 22.11.2018) Raspberry Pi 2 Model B (Reduced Schematics) URL: https://www.raspberrypi.org/documentation/hardware/raspberrypi/schematics/rpi_SCH_2b_1p2_reduced.pdf

[19] Raspberry Pi Foundation (luettu 10.8.2018) Raspberry Pi OS URL: https://www.raspberrypi.org/documentation/raspbian/

[20] The Apache Software Foundation (luettu 10.8.2018) What is the Apache HTTP Server Project? URL: https://httpd.apache.org/ABOUT_APACHE.html

[21] HTML 5.3 Editor's Draft 13 September 2018 (luettu 10.8.2018) URL: https://w3c.github.io/html/sec-forms.html#sec-forms

[22] Copyright © 1997 - 2018 by the PHP Documentation Group. (luettu 25.11.2018), $_POST URL: http://php.net/manual/en/reserved.variables.post.php

[23] Copyright © 1997 - 2018 by the PHP Documentation Group. (luettu 10.8.2018) shell_exec() URL: http://php.net/manual/en/function.shell-exec.php

[24] Copyright © 1997 - 2018 by the PHP Documentation Group. (luettu 10.8.2018) escapeshellcmd() URL: http://php.net/manual/en/function.escapeshellcmd.php

[25] Ben Croston, 2018 Python Software Foundation, (luettu 10.8.2018) RPi.GPIO 0.6.3 URL: https://pypi.org/project/RPi.GPIO/

9. LIITTEET

Liite 1. Kellon ohjaussivun HTML-koodi
Liite 2. Moottoreiden kalibrointi koodi
Liite 3. Herätysajan PHP-koodi
Liite 4. Herätysajan Python-koodi
Liite 5. Kesä/Talvi-ajan PHP-koodi
Liite 6. Kesä/Talvi-ajan Python-koodi

Liite 1. Kellon ohjaussivun HTML-koodi

```
<html>
   <head>
      <title>Time control page</title>
   </head>
   <body>
      <img src = "hauska.gif" alt = "GIF here" align = "center">

      <form action = change_alarm.php method = "post">
         <center>
            New alarm:
            <select name ="new_alarm">
               <option value = "1" > 01:00/13:00
               <option value = "2" > 02:00/14:00
               <option value = "3" > 03:00/15:00
               <option value = "4" > 04:00/16:00
               <option value = "5" > 05:00/17:00
               <option value = "6" > 06:00/18:00
               <option value = "7" selected> 07:00/19:00
               <option value = "8" > 08:00/20:00
               <option value = "9" > 09:00/21:00
               <option value = "10" > 10:00/22:00
               <option value = "11" > 11:00/23:00
               <option value = "12" > 12:00/00:00
            <input type ="submit" value ="Enter">
            </select>
         </center>
      </form>
      <form action = "change_time.php" method = "post">
         <p align = "right">
            Aika: <br>
            <input type = "radio" name = "time" value = "summer"> Kes&auml <br>
            <input type = "radio" name = "time" value = "winter" checked>Talvi <br>
            <input type = "submit" value = "Enter">
         </p>
      </form>
      <form action = "change_time.php" method = "post">
         <p align = "left">
            <h1>Liikuta viisareita</h1><br>
            Eteenp&aumlin Taaksep&aumlin<br>
            <input type = "submit" name = "forward" value = "Eteenpäin"> <input
type = "submit" name = "backwards" value = "Taaksepäin">

      </form>

   </body>
</html>
```

Liite 2. Ajan kalibrointikoodi

```
import time
import sys
import RPi.GPIO as GPIO

#HIGH = LDR not under light
#LOW = LDR direct light contact

#If RasPi's internal pull up resistor is used -> comparator output will work
#in scale [0,3.3]V. Other option external pull-up with voltage reference from
#3.3V regulator

#########################
GPIO.setmode(GPIO.BOARD)
GPIO.setup(18, GPIO.OUT)
GPIO.setup(22, GPIO.IN)
signal = GPIO.PWM(18, 50)
#########################
dc_forward = 7.75



print("Calibration test is on")
#Poll comparator input so we can define is the clock on time
try:
   while True:
      if GPIO.input(22):
         #print("Input was HIGH and clock is on time")
         signal.stop()
      else:
         #print("Input was LOW, clock is NOT on time")
         signal.start(dc_forward)
except KeyboardInterrupt:
   GPIO.cleanup()
   print("CLEAR AND END")

GPIO.cleanup()
print("CLEAR")
```

Liite 3. Herätysajan PHP-koodi

```
<?php
$new = $_POST["new_alarm"];

$script = escapeshellcmd("python3 /var/www/html/change_alarm.py {$new}");

$execute = shell_exec($script);
echo $execute;
echo"test";
?>
```

Liite 4. Herätysajan Python-koodi

```
import sys
import time
import RPi.GPIO as GPIO

new_alarm = sys.argv[1]
#print("New alarm time is", new_alarm)


GPIO.setmode(GPIO.BOARD)
GPIO.setup(16, GPIO.OUT)
signal = GPIO.PWM(16, 50)


def define_duty_cycle(new_alarm):

    dc_increase_per_hour = int((2300 - 500) / 12)
    new_dc = (dc_increase_per_hour * int(new_alarm) + 500) #New duty cycle in
microseconds
    #pwm method wants % as[0.0, 100.0]
    new_dc_percentage = float((new_dc / 20000)*100)
    rounded_dc = round(new_dc_percentage, 1)

    return rounded_dc

""main""

###Give time for GPIO to instantiate
time.sleep(4)

new1 = define_duty_cycle(new_alarm)
#print("Duty cycle is: % ", new1)
time.sleep(2)

signal.start(new1)
time.sleep(4)

signal.stop()
GPIO.cleanup()
print("Clean up")
```

Liite 5. Kesä/Talvi-ajan PHP-koodi

```
<?php
$time = $_POST["time"];
$forward = $_POST["forward"];
$backwards = $_POST["backwards"];
$arg = "";

$script = escapeshellcmd("python3 /var/www/html/change_time.py
{$time}{$forward}{$backwards}");
$execute = shell_exec($script);
echo $execute;

?>
```

Liite 6. Kesä/Talvi-ajan Python-koodi

```
import RPi.GPIO as GPIO
import time
import sys



GPIO.setmode(GPIO.BOARD)
GPIO.setup(18, GPIO.OUT)#360-servo control pin
signal = GPIO.PWM(18, 50)

time.sleep(3)

param = sys.argv[1]
time_file = open("time_state.txt", "r")
file = time_file.read()
current_time = file.strip("\n")
time_file.close()
print("Current timestate was",current_time)

dc_forward = 1550
dc_backward = 1400
T = 20000
dc_forward_p = 6.5
dc_backward_p = 7.75
#Eteenpin
#Taaksepin
#summer
#winter


'"main"'

if (param == "summer" and current_time == "winter"):
    time_file = open("time_state.txt", "w")
    time_file.write("summer")
    time_file.close()
    signal.start(dc_forward_p)
    time.sleep(4)
    signal.stop()
    print("winter -> summer")
elif (param == "winter" and current_time == "summer"):
    time_file = open("time_state.txt", "w")
    time_file.write("winter")
    time_file.close()
    signal.start(dc_backward_p)
    time.sleep(4)
```

```
    signal.stop()
    print("summer -> winter")


elif (param == "Eteenpin"):
    signal.start(dc_forward_p)
    time.sleep(4)
    signal.stop()

elif (param == "Taaksepin"):
    signal.start(dc_backward_p)
    time.sleep(4)
    signal.stop()

else:
    print("Nothing to do")

time_setting = param
GPIO.cleanup()
print("PINS CLEANED")
```