Tampereen yliopisto

Alexandre Ignatiev

PARISTO- TAI AKKUKÄYTTÖINEN ITSENÄISESTI TOIMIVA INFONÄYTTÖ LANGATTOMAAN LÄHIVERKKOON

Diplomityö
Informaatioteknologian ja viestinnän tiedekunta
Tarkastajat: Yliopistonlehtori Erja Sipilä
Professori Karri Palovuori
Toukokuu 2020

TIIVISTELMÄ

Alexandre Ignatiev: Paristo- tai akkukäyttöinen itsenäisesti toimiva infonäyttö langattomaan lähiverkkoon
Diplomityö
Tampereen yliopisto
Sähkötekniikan DI-tutkinto-ohjelma
Toukokuu 2020

Tässä työssä toteutettiin paristo- tai akkukäyttöinen itsenäisesti toimiva infonäyttö langattomaan lähiverkkoon. Tavoitteena oli toteuttaa laite, jonka näytöllä olevat tiedot ovat jatkuvasti luettavissa. Lisäksi laitteen tuli olla edullinen ja suhteellisen helppo toteuttaa.

Työn toteutukseen valittiin sähköpaperinäyttö. Tämä mahdollisti paristokäyttöisen laitteen toteuttamisen niin, että sen esittämä informaatio on jatkuvasti luettavissa. Ohjelmoitavaksi komponentiksi työhön valittiin WiFi-yhteyksillä varustettu ESP32-piiri. Laite toteutettiin niin, että sen energiavarastona voivat toimia joko alkaliparistot tai litiumioniakku, jotta näiden soveltuvuutta voidaan verrata. Lisäksi työstä tehtiin sekä lineaariregulaattorilla että hakkuriregulaattorilla varustettu versio.

Komponenttivalinnan jälkeen laitteelle suunniteltiin kytkentä ja siitä toteutettiin prototyyppi. Prototyyppiä varten valmistutettiin myös piirilevy, mutta sen tarkoitus oli vain helpottaa laitteen toteuttamista eikä sitä käsitellä tässä työssä tarkemmin. Tämän jälkeen toteutettu prototyyppi ohjelmoitiin hakemaan kalenteritietoja Google Calendar -palvelusta 15 minuutin välein ja esittämään nämä näytöllään.

Prototyypin rakentamisen jälkeen mitattiin näytön päivittämiseen kuluva energia ja arvioitiin laitteen toiminta-aika valituilla paristoilla tai akulla. Lisäksi työssä arvioitiin lineaariregulaattorin ja hakkuriregulaattorin vaikutusta molempien energialähteiden tapauksessa.

Työssä saatiin toteutettua toimiva laite, joka hakee kalenteritapahtumat Google Calendar -palvelusta ja esittää ne näytöllään. Laitteen toiminta-ajaksi arvioitiin puolitoista vuotta, kun käytössä on kolme AA-koon alkaliparistoa. Tämä arvio tehtiin olettaen, että laite päivittyy 15 minuutin välein ja ainoastaan tiettyyn aikaan päivästä. Lisäksi työssä todettiin, että lineaariregulaattori on toimiva valinta akkua käytettäessä. Paristojen tapauksessa sen sijaan kannattaa käyttää hakkuriregulaattoria ja useampaa paristoa sarjassa.

Työn tuloksena voidaan myös todeta, että laite soveltuu muihinkin käyttötarkoituksiin, joissa esitettävän informaation määrä on suhteellisen pieni ja päivitystahti on suhteellisen hidas. Laite voisi esimerkiksi esittää uutisotsikoita, muistilistoja, osakekursseja tai vaikkapa säätietoja.

Avainsanat: ESP32, WiFi, sähköpaperi, esineiden internet, paristokäyttöinen, sähköinen kalenteri

Tämän julkaisun alkuperäisyys on tarkastettu Turnitin OriginalityCheck -ohjelmalla.

ABSTRACT

Alexandre Ignatiev: Battery powered standalone infodisplay for wireless networks
Master's Thesis
Tampere University
Degree Programme in Electrical Engineering, MSc (Tech)
May 2020

The main objective of this thesis was to design a battery powered standalone infodisplay with wireless connectivity. Additional objectives were to make the device relatively inexpensive and easy to implement. The information should also be constantly displayed on the device.

Electronic paper display was chosen for the implementation of the device, which made it possible for the battery powered device to be always readable and have a decent battery life. ESP32-module was chosen as a WiFi-radio and a CPU for the device. The device was designed to work with both alkaline bateries and lithium ion cells so that their suitability for this device could be compared. The implemented device can also include either a linear or a buck-boost regulator.

A schematic and a prototype were devised after the component selection. A PCB design was also created from the schematic to ease the development of the device. However the PCB design is not the main topic of this thesis. Two prototypes of the device were built, one including the linear regulator and one including the buck-boost converter. The prototypes were programmed to fetch and display events from Google Calendar every 15 minutes.

After the prototype was complete, the current consumption of the device was measured and the battery life of the device was estimated in case of both alkaline and lithium ion batteries. Additionally the effects of the linear and buck-boost regulators on functionality of the device were investigated.

The result of this thesis is a working device which fetches events from Google Calendar and displays them on the electronic paper display. The battery life of the device was estimated to be one and a half years assuming three AA-batteries are used and limiting the display updates to every 15 minutes and only at specific time interval during the day. Additional conclusion was that the linear regulator is suitable to be used with the lithium ion battery. With the alkaline batteries however, it's recommended to use a buck-boost converter because of the discharge voltage curve of the batteries.

As a result of this thesis, it's also safe to say that this device could be used for similar applications, which require only occasional updates and have relatively low amount of info to display. Such applications could include news topics, notes, stocks and weather info.

Keywords: ESP32, WiFi, Electronic Paper, IoT, Battery-powered, iCalendar

The originality of this thesis has been checked using the Turnitin OriginalityCheck service.

ALKUSANAT

Tässä työssä toteutettu laite oli oma ideani ja toteutin sen kokonaan itse. Laitteen tausta liittyy haluun saada informaatiota ilman, että sitä varten joutuu käynnistämään perinteistä tietotekniikkaa kuten tietokonetta, puhelinta, tablettia taikka televisiota. Tämä laite on jatkumo kandidaatintyölleni, jossa infonäyttö oli toteutettuna perinteisellä LCD-näytöllä ja ESP8266-moduulilla. Paristokäyttöisen laitteen ja sen vaatiman sähköpaperinäytön käyttäminen oli luonnollinen seuraava askel.

Vastaavia projekteja on olemassa Internetissä useita muitakin. Tämän työn toteutukseen ei kuitenkaan otettu vaikutteita muista projekteista, mikä näkyy ehkä vastaan tulleiden ongelmien määrästä. Virheistä opittujen asioiden seurauksena mielestäni laitteesta saa kuitenkin melko pienillä parannuksilla ihan hyvän tuotteen ja se on minulle tärkeintä.

Työn tekeminen oli mielekästä kun käytössä olivat sopivat fyysiset työkalut ja ohjelmistot. Opin myös tämän työn aikana korvaamaan ne työkalut, jotka eivät olleet sopivia. Työn toteutusta helpottivat huomattavasti Visual Studio Code -koodieditori, EasyEDA-työkalu ja Overleaf-editori.

Haluan tässä kiittää Tampereen yliopiston puolesta toimineita työn tarkastajia Karri Palovuorta ja Erja Sipilää. Haluan myös kiittää heitä sekä muun muuassa Jouko Heikkistä ja Matti Rintalaa hyvin toteutetuista ja mielenkiintoisista kursseista, joilla oli merkittävä vaikutus tämän työn onnistumiseen. Lisäksi haluan kiittää tyttöystävääni ja ystäviäni, jotka tukivat vaikeina hetkinä.

Tampereella, 18. toukokuuta 2020

Alexandre Ignatiev

SISÄLLYSLUETTELO

1 Johdanto . . . 1
2 Komponenttien ja tekniikoiden valinta . . . 2
2.1 Kytkennät . . . 2
2.1.1 Näyttö . . . 2
2.1.2 Sähköpaperi . . . 4
2.1.3 WiFi-moduuli . . . 6
2.1.4 Energiavarasto . . . 7
2.1.5 Jännitemuunnos . . . 8
2.1.6 Oheiskytkennät . . . 10
2.2 Ohjelmisto . . . 11
2.2.1 Langaton lähiverkko . . . 11
2.2.2 Kellonaika . . . 12
2.2.3 Kalenteritiedon haku . . . 13
2.2.4 Näytön päivittäminen . . . 16
2.2.5 Unitila . . . 17
2.2.6 Asetukset . . . 18
3 Laitteen toteutus . . . 20
3.1 Kytkentä . . . 20
3.1.1 Näyttö . . . 20
3.1.2 Energiavarasto . . . 21
3.1.3 Regulaattori . . . 22
3.1.4 ESP32 . . . 23
3.1.5 Oheiskytkennät . . . 25
3.1.6 Virrankulutus unitilassa . . . 26
3.2 Ohjelmisto . . . 26
3.2.1 Ylemmän tason toiminta . . . 27
3.2.2 Kalenteritietojen haku . . . 28
3.2.3 Asetukset . . . 29
3.2.4 Unitila . . . 29
3.3 Toiminnan optimointi . . . 30
3.3.1 Verkkoyhteyksien optimointi . . . 30
3.3.2 Näytön optimointi . . . 32
3.3.3 Muut optimoinnit . . . 35
4 Toteutuksen analysointi . . . 36
4.1 Lopullinen prototyyppi . . . 36
4.2 Laitteen toiminta-aika . . . 37

4.3 Virrankulutus unitilassa 39
4.4 Hinta 39
4.5 Hakkuriregulaattorin ongelmat 40
5 Yhteenveto 42
5.1 Käytettävyys 42
5.2 Paristo- tai akkukäyttöisyys 43
5.3 Regulaattorivalinnat 43
5.4 Laitteen ongelmat 44
5.5 Parannusehdotukset 44
5.6 Käytetyt työkalut 45
Lähteet 47

KUVALUETTELO

2.1 Eri näyttöteknologioiden virrankulutukset. 3
2.2 Sähköpaperinäytön rakenne [10]. 5
2.3 Alkaliparistoja litiumioniakun jännitekäyrät [12]. 9
2.4 Esimerkki 8x5 kokoisesta, tasalevyisestä kirjasintyypistä. 16
2.5 Form-avainsanan avulla luotu HTML-lomake. 18

3.1 Sähköpaperinäytön hakkurikytkentä. 21
3.2 Lineaariregulaattorin kytkentä. 22
3.3 Hakkuriregulaattorin kytkentä. 23
3.4 ESP32-piirin kytkentä. 24
3.5 Energiavaraston jännitteenmittauskytkentä. 25
3.6 Litiumioniakun latauspiirin kytkentä. 25
3.7 Tilakaavio ohjelman ylemmän tason toiminnasta. 27
3.8 Kalenteritietojen haun tilakaavio. 28
3.9 Sähköpaperinäytön nopean päivityksen seurauksena näkyvä haamukuva. . 33

4.1 Kuva toteutetun laitteen piirilevystä. 36
4.2 Kuva toteutetun laitteen näytöstä, kun käytössä on pieni kirjasintyyppi. . . . 37
4.3 Kuva toteutetun laitteen näytöstä, kun käytössä on iso kirjasintyyppi. 37
4.4 Optimoidun laitteen virrankulutus, kun näyttö päivitetään nopeasti. 38
4.5 Optimoidun laitteen virrankulutus, kun näyttö päivitetään kokonaan. 38

TAULUKKOLUETTELO

2.1 Eri näyttöteknologioiden vertailu [4][5][6]. 3
2.2 Sähköpaperinäyttöjen eri kokojen vertailu [7][8][9]. 4
2.3 ESP32 moduulien ominaisuuksien vertailu [14]. 7
2.4 Kalenteritiedon hakuvaihtoehdot [32][33][36]. 14
2.5 ESP32-piirin unitilat [13]. 17

3.1 Laitteen komponenttien virrankulutukset unitilassa. 26

4.1 Laitteessa käytettyjen oleellisten komponenttien hinnat. 40

OHJELMA- JA ALGORITMILUETTELO

2.1 curl-ohjelman tuloste kalenteritietojen haussa. 15
2.2 Form-avainsanan avulla luotu HTML-lomake. 18
3.1 ESP32-ohjelman debug tulosteet ilman optimointia. 31
3.2 ESP32-ohjelman debug-tulosteet verkko-optimointien kanssa. 32
3.3 Näytön päivittämiseen liittyvät debug-tulosteet ilman optimointia. 33
3.4 Näytön päivittämiseen liittyvät debug-tulosteet optimointien kanssa. 34

LYHENTEET JA MERKINNÄT

ADC	Analogia-digitaalimuunnin (engl. Analog to Digital Converter)
API	Ohjelmointirajapinta (engl. Application Programming Interface)
BLE	Langaton tietoliikennetekniikka (engl. Bluetooth Low Energy)
ChLCD	LCD-näytön variaatio (engl. Cholesteric Liquid Crystal)
CSV	Taulukkoformaatti (engl. Comma Separated Value)
DC	Signaali, jolla määritellään onko lähetettävä informaatio kuva- vai ohjausdataa (engl. Data / Command)
DMA	Muistiosoitusmenetelmä (engl. Direct Memory Access)
DNS	Internetin nimipalvelujärjestelmä (engl. Domain Name System)
DPI	Pistettä tuumaa kohti (engl. Dots Per Inch)
EN	Toimminnaltaan reset-pinniä vastaava pinni ESP32-piirissä (engl. Enable)
EPD	Sähköpaperinäyttö (engl. Electrophoretic Display)
FET	Kanavatransistori (engl. Field Effect Transistor)
FreeRTOS	Reaaliaikakäyttöjärjestelmätoteutus (engl. Real Time Operating System)
GDR	Sähköpaperinäytön signaali (engl. Gate Drive)
GET	HTTP-protokollan kysely
HEAD	HTTP-protokollan kysely
HTTP	Hypertekstin siirtoprotokolla (engl. Hyper Text Transfer Protocol)
IO	Yleiskäyttöinen pinni (engl. Input Output)
IP	TCP/IP-viitemallin verkkokerroksen protokolla (engl. Internet Protocol)
ITO	Indiumtinaoksidi, sähköä johtava läpinäkyvä materiaali (engl. Indium Tin Oxide)
JLCPCB	Piirilevyvalmistaja (JiaLiChuang (HongKong) Co., Limited)
JTAG	Ohjelmiston ja laitteiston testaukseen käytettävä portti (engl. Joint Test Action Group)
LCD	Nestekidenäyttö (engl. Liquid Crystal Display)
LCSC	Elektroniikkakauppa (Shenzhen LCSC Electronics Technology Co., Ltd.)

LED	Hohtodiodi (engl. Light Emitting Diode)
OLED	Orgaaniseen aineeseen perustuva näyttötyyppi (engl. Organic Light Emitting Diode)
PREVGH	Sähköpaperinäytön signaali (engl. Unregulated High Voltage)
PREVGL	Sähköpaperinäytön signaali (engl. Unregulated Low Voltage)
PSRAM	Muistityyppi (engl. Pseudo Static Random Access Memory)
RAM	Muistityyppi (engl. Random Access Memory)
RESE	Sähköpaperinäytön signaali (engl. Resistor Sense)
RFC	Internetiä koskevia standardeja (engl. Request for Comments)
RTC	Reaaliaikakello (engl. Real Time Clock)
RXD	Sarjaväylän vastaanottosignaali (engl. UART Receive)
SNTP	Protokolla aikatiedon välittämiseen (engl. Simple Network Time Protocol)
SoC	Mikropiirityyppi (engl. System on Chip)
SPI	Sarjaväylä (engl. Serial Peripheral Interface)
SSID	Langattoman lähiverkon verkkotunnus (engl. Service Set Identifier)
SSL	Salausprotokolla (engl. Secure Sockets Layer)
TCP	Tietoliikenneprotokolla (engl. Transmission Control Protocol)
TFT	Ohutkalvotransistori (engl. Thin Film Transistor)
TLS	Salausprotokolla (engl. Transport Layer Security)
TME	Elektroniikkakauppa (Transfer Multisort Elektronik sp. z o.o.)
TXD	Sarjaväylän lähetyssignaali (engl. UART Transmit)
ULP	Pienitehoinen prosessoriydin ESP32-piirissä (engl. Ultra Low Power)
UTF	Merkistöstandardi (engl. Unicode Transformation Format)
WPA	Salausprotokolla (engl. Wi-Fi Protected Access)

1 JOHDANTO

Nykyään on olemassa monenlaisia sähköisiä palveluita kuten kalentereita, sähköposteja ja tehtävälistoja, jotka voidaan synkronoida kätevästi eri laitteiden välillä. Synkronoinnin lisäksi nämä palvelut mahdollistavat muun muuassa yhteiskäytön, eli useampi taho voi muuttaa esimerkiksi jaettua tehtävälistaa. Tällaisten palveluiden käyttö vaatii kuitenkin jonkin laitteen, esimerkiksi tietokoneen, puhelimen tai tabletin. Näitä laitteita ei voi kuitenkaan hintansa ja virrankulutuksensa vuoksi käyttää esimerkiksi seinäkalenterina. Markkinoilla ei myöskään ole mitään erillistä laitetta, joka soveltuisi hyvin pelkästään sähköisistä palveluista haettavan tiedon esittämiseen.

Tämän työn tarkoituksena on suunnitella laite, joka yhdistää edellä mainittujen sähköisten palvelujen sekä esimerkiksi jääkaapin ovessa olevan tehtävälistan tai seinäkalenterin hyvät puolet. Laitteen on tarkoitus olla yhteydessä langattomaan Internet-verkkoon. Sen on oltava aina päällä ja sen on toimittava paristojen tai akun varassa, jotta sen voi sijoittaa ilman johtojen tuomia rajoitteita. Lisäksi laitteen on oltava suurimmalta osin autonominen, joten sen käyttöliittymä voi olla minimaalinen. Laitteen täytyy olla myös edullinen ja suhteellisen yksinkertainen toteuttaa.

Tämän lisäksi laitteen on tarkoitus käyttää mahdollisimman paljon olemassa olevia järjestelmiä ja palveluita. Langattoman verkon tapauksessa tämä tarkoittaa WiFi-verkkoa, sillä sen voi olettaa olevan saatavilla kaikkialla, missä kyseistä laitetta haluttaisiin käyttää. Tässä työssä toteutetaan Google Calendar -palveluun yhdistyvä infonäyttö. Kyseisen infonäytön tarkoituksena on esittää verkosta haettavien kalenteritapahtumien tiedot. Tätä laitetta voisi käyttää muihinkin tarkoituksiin, esimerkiksi edellämainitun tehtävälistan tai vaikkapa säätietojen esittämiseen. Tässä työssä keskitytään kuitenkin kalenterin toteuttamiseen ja loput mahdolliset käyttötarkoitukset käsitellään lyhyesti ja vain teoriatasolla.

Luvussa 2 käsitellään elektroniikka- ja ohjelmistokomponenttien valinta. Tässä vaiheessa tehdään karkea komponenttien valinta ja käsitellään niiden käyttöön liittyvä teoria. Lisäksi tässä käsitellään vaihtoehtoisia tapoja laitteen käytännön toteutukseen. Luvussa 3 käsitellään laitteen toteutus. Tämä sisältää sekä elektroniikan että ohjelmiston. Lisäksi tässä luvussa käsitellään mahdollisia optimointeja, jotka parantavat laitteen käytettävyyttä ja toiminta-aikaa. Luvussa 4 esitetään yhteenveto tästä työstä.

2 KOMPONENTTIEN JA TEKNIIKOIDEN VALINTA

Tässä työssä kehitettävä laite on sulautettu järjestelmä eli sähköinen laite, jossa on jokin ohjelmoitava osa. Laitteen on tarkoitus olla paristo- tai akkukäyttöinen ja sillä on oltava mahdollisimman pitkä toiminta-aika. Tämä on otettava huomioon sekä elektroniikan että ohjelmiston suunnittelussa. Mahdollisimman pitkän toiminta-ajan saavuttamiseksi laitteen on oltava suurimman osan ajasta lepotilassa ja herättävä tästä lepotilasta vain lyhyeksi ajaksi näytön päivittämistä varten [1].

2.1 Kytkennät

Suunniteltavan laitteen päätarkoitus on esittää informaatiota, joten laitteen tärkein osa on siinä käytettävä näyttö. Tämän lisäksi laitteeseen tarvitaan näyttöä ohjaava prosessori tai mikrokontrolleri sekä jokin WiFi-yhteydet mahdollistava osa. Edellä mainittujen osien vaatimuksista ja laitteen paristokäyttöisyydestä johtuen kytkentään tarvitaan myös jokin jännitettä reguloiva osa. Lisäksi tarvitaan myös muita komponentteja, kuten painonappeja käyttöliittymää varten, ohjelmointiliittimiä, merkkivaloja sekä laitteen kehittämistä helpottavia komponentteja.

2.1.1 Näyttö

Näytön valinnassa on ensiksi otettava huomioon, että eri näyttöteknologioiden virrankulutuksessa on huomattavia eroja [2]. Laitteen paristo- tai akkukäyttöisyyden takia sen virrankulutus halutaan minimoida toiminta-ajan maksimoimiseksi.

Näyttöteknologiat voidaan karkeasti jakaa valoa tuottaviin näyttöihin (emissive) sekä sellaisiin, jotka tarvitsevat ulkoisen valonlähteen (non-emissive). Esimerkkinä ensimmäisestä on OLED-näyttö (Organic Light Emitting Diode), jossa jokainen pikseli toimii valonlähteenä. Esimerkkinä jälkimmäisestä on EPD-näyttö (Electrophoretic Display) eli sähköpaperi, joka ei tuota valoa ja on luettavissa ulkoisen valaistuksen avulla kuten tavallinen paperi[1]. Poikkeuksena tästä jaottelusta on esimerkiksi LCD-näyttöteknologia (Liquid Crystal Display), josta on olemassa versioita sekä taustavalolla että ilman. Näistä ensimmäisen kategorian laitteet kuluttavat huomattavasti enemmän energiaa johtuen siitä, että ne tuottavat valoa tai tarvitsevat välttämättä taustavalon. Tästä johtuen suunniteltavassa

[1] Sähköpaperista on olemasta versio, jossa näytön edessä on ohut läpinäkyvä kalvo, joka valaisee näyttöä päältäpäin [3].

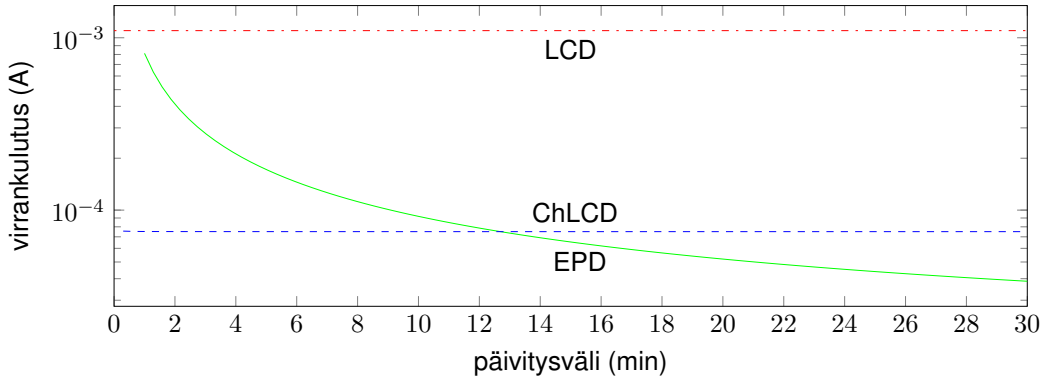
laitteessa onkin järkevää käyttää näyttöä, joka on luettavissa vain ulkoisen valonlähteen avulla. [2]

Toinen näyttöteknologioiden jaotteluperuste on kuvan pysyvyys. Suurimmassa osassa näyttöteknologioista kuva ei tallennu näyttöpaneeliin, vaan sitä täytyy päivittää jatkuvasti, jotta se näkyisi katsojalle. Poikkeuksena tästä ovat ainakin edellä mainittu sähköpaperi sekä ChLCD-näyttö (Cholesteric Liquid Crystal) eli eräs LCD-näytön variaatio. Tämän tyyppiset näytöt perustuvat johonkin bistabiiliin ilmiöön ja tarvitsevat energiaa ainoastaan kuvan muuttamiseen. Nämä ovat melko uusia näyttötyyppejä, joten niiden ongelmana on suhteellisen iso hinta. Sitäkin isompi ongelma on, että Cholesteric liquid crystal -näyttöä on vaikea hankkia Euroopasta. [2]

Taulukko 2.1. *Eri näyttöteknologioiden vertailu [4][5][6].*

Näyttö	Tyyppi	Koko	Resoluutio	Virrankulutus (unitila)	Hinta (€)
RX240128A-FGN	LCD	4,17"	240x128	1,1mA	18
DEE 400300A2-W	EPD	4,2"	400x300	15mA (2uA)	30
LS044Q7DH01	ChLCD	4,4"	320x240	200uA (75uA)	70

Edellä mainittujen tietojen perusteella voidaan vertailla tarkemmin LCD-näyttöä ilman taustavaloa, sähköpaperinäyttöä sekä ChLCD-näyttöä. Valitaan LCD-näytöksi Raystar Optronicsin valmistama RX240128A-FGN [4], sähköpaperinäytöksi Display Elektronikin valmistama DEE 400300A2-W [5] ja ChLCD-näytöksi Sharpin valmistama LS044Q7DH01 [6]. Näiden oleelliset ominaisuudet on esitetty taulukossa 2.1. Taulukosta voidaan nähdä, että valittujen näyttöjen fyysiset koot ovat samaa luokkaa, mutta virrankulutukset ja hinnat poikkeavat huomattavasti. Taulukosta voidaan havaita myös, että LCD-näytöllä ei ole muihin näyttöihin verrattavaa unitilaa, koska näyttöä täytyy päivittää jatkuvasti, jos haluaa, että näytön kuva ei häviä.



Kuva 2.1. *Eri näyttöteknologioiden virrankulutusvertailu.*

Johtuen siitä, että näyttöjen virrankulutus lepotilassa ja aktiivisessa tilassa on eri, täytyy tilannetta tarkastella myös näyttöjen eri päivitysvälin suhteen. Tämä on esitetty kuvassa 2.1. Kuvaajasta nähdään, että LCD-näyttö kuluttaa huomattavasti enemmän virtaa muihin teknologioihin verrattuna, kunhan päivitysväli on tarpeeksi pitkä. Kuvaajasta voi myös nähdä, että ChLCD-näyttö olisi paras vaihtoehto, jos päivitysväli olisi alle 15 minuuttia.

Infonäyttöä ei kuitenkaan ole tarvetta päivittää niin usein vaan päivitysväli voi olla tilanteesta ja käyttötarkoituksesta riippuen huomattavasti tätäkin pidempi.

Näytön valinnalle on muitakin kriteerejä, esimerkiksi fyysinen koko ja resoluutio. Näytön esittämää kuvaa on yleensä tarkoitus pystyä lukemaan tietyltä etäisyydeltä, mikä määrittelee esimerkiksi näytöllä esitettävän tekstin minimikoon. Näytön fyysinen koko määrää, paljonko informaatiota voidaan esittää näytöllä. Näytön resoluutio sen sijaan määrittelee, paljonko kuvapisteitä (pikseleitä) näytössä on pysty- ja vaakasuunnassa, eli käytännössä, mikä on pienimmän näytöllä esitettävissä olevan yksityiskohdan koko. Resoluutio määrittelee siis kuinka yksityiskohtaisen näytöllä esitettävästä kuvasta tai tekstistä saa. Lisäksi se rajoittaa pienimmän mahdollisen esitettävissä olevan kirjasintyypin koon, jos katselumukavuutta ei oteta huomioon.

Taulukko 2.2. *Sähköpaperinäyttöjen eri kokojen vertailu [7][8][9].*

Läpimitta (tuuma)	Koko (mm) leveys x korkeus	Resoluutio pysty x vaaka	DPI-arvo
2,9	80 x 36	296 x 128	112
4,2	90 x 77	400 x 300	120
7,5	170 x 111	640 x 384	100

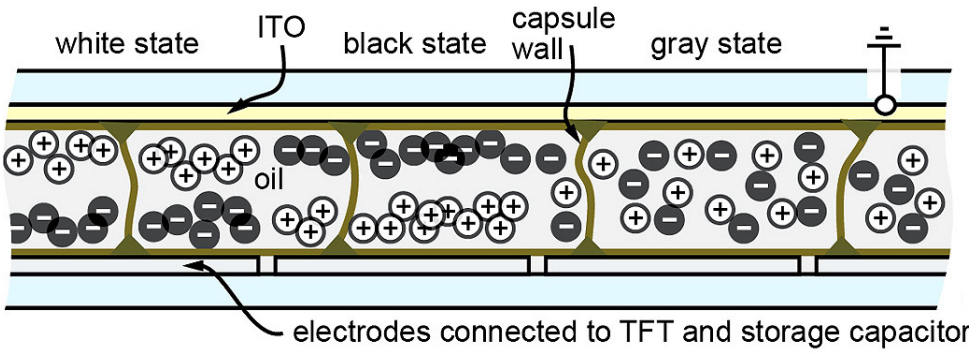
Näytön fyysinen koko ja näytön resoluutio ovat kuitenkin useimmiten sidottuina toisiinsa. Näytön isompi fyysinen koko tarkoittaa isompaa resoluutiota. Tätä on järkevämpi tarkastella näytön DPI-arvon (Dots Per Inch) avulla. Taulukossa 2.2 on esitetty Waveshare Electronicsin valmistamien näyttöjen fyysiset koot, resoluutiot ja DPI-arvot. Taulukosta voidaan havaita, että jokaista näyttökokoa vastaa jokin tietty resoluutio ja että DPI-arvojen vaihtelu on hyvin pientä.

2.1.2 Sähköpaperi

Sähköpaperi on näyttötyyppi, joka perustuu elektroforeesiin eli sähköisesti varautuneiden hiukkasten liikkeeseen sähkökentässä. Näytössä on pieniä soluja, joissa on sähköisesti varautuneita kahden eri värin hiukkasia. Solujen molemmin puolin on elektrodeja, joilla soluihin saa luotua sähkökentän. Sopivalla sähkökentällä voi valita kummat solun värihiukkasista siirtyvät näytön etupuolelle ja peittävät toisen taakse jäävän värin. Näytön elektrodit on asetettu matriisin muotoon, jotta näytön eri kohdissa voi olla eri väri eli näyttöön saa kuvan. Soluissa on värihiukkasten lisäksi eristeenä toimivaa öljyä, joka varmistaa, että värihiukkaset liikkuvat hallitusti eivätkä liiku soluissa ilman ulkoista sähkökenttää. Yleensä näytön värihiukkaset ovat valkoisia ja mustia, jotta sähköpaperi muistuttaisi tavallista paperia, jolle on tulostettu mustaa tekstiä. [10]

Kaksivärisen sähköpaperinäytön rakenne on esitetty kuvassa 2.2. Kuvassa näkyvät *white state* ja *black state* ilmaisevat näytöllä näkyvää valkoista ja mustaa väriä, kun vastaavan värin hiukkaset ovat lähellä näytön pintaa. *Gray state* sen sijaan ilmaisee näiden välitilaa, jossa pinnan lähellä on sekä valkoisia että mustia hiukkasia. Kuvassa esitetty *capsule*

wall tarkoittaa kunkin hiukkasia sisältävän solun seinämää. Lisäksi kuvassa on esitetty *ITO* (Indium Tin Oxide), joka tarkoittaa näytön pinnalla olevaa läpinäkyvää ja sähköä johtavaa pintaa [2]. Kuvassa on esitetty myös näytön alareunassa näkyvät elektrodit, jotka on kytketty niitä ohjaaviin TFT-transistoreihin (Thin Film Transistor). [10]



Kuva 2.2. *Sähköpaperinäytön rakenne [10].*

On olemassa myös kolmivärisiä sähköpaperinäyttöjä, joissa kolmas väri on esimerkiksi punainen. Tällaisten näyttöjen toiminta perustuu siihen, että kolmannen värin hiukkaset ovat isompia kuin kaksi muuta väriä. Kolmannen värin hiukkasilla on sama varaus kuin esimerkiksi mustan värin hiukkasilla, mutta kokonsa takia ne liikkuvat huomattavasti hitaammin kuin mustat hiukkaset. Tämä tarkoittaa sitä, että sopivalla sähkökentän ajoituksella voidaan valita, kummat näistä saman varauksen omaavista hiukkasista päätyvät näytön pintaan. Sähköpaperinäytön kuvanlaatu kärsii kuitenkin kolmannen värin lisäämisestä ja sen päivitysaika on myös huomattavasti pidempi. [11]

Sähköpaperin toimintamekanismin seurauksena näytön kuva säilyy ilman ulkoista sähkökenttää. Haittapuolena näytön kuvan päivittäminen on suhteellisen hidasta muihin näyttötekniikoihin verrattuna. Kuvan säilyvyyden tuoma lisäetu on, että koko kuvaa ei tarvitse päivittää kerralla. Kuvan päivittäminen osittain aiheuttaa kuitenkin haamukuvia, joiden poistamiseksi näyttö täytyy säännöllisesti tyhjentää kokonaan. Nämä haamukuvat eivät kuitenkaan estä näkemästä päivitettyä kuvaa. [10]

Pelkän näyttöpaneelin eli elektrodimatriisin jokaisen kuvapisteen ohjaaminen yksitellen vaatisi erittäin monta johdinta, joten näyttöihin on tapana integroida ohjainpiiri. Tämä ohjain sisältää sarja- tai rinnakkaisväylän, jonka avulla se kommunikoi mikroprosessorin kanssa. Lisäksi siinä on muistia kuvan tallentamista varten sekä tarvittavaa elektroniikkaa edellä mainitun elektrodimatriisin ohjaamiseen. Yleinen väylä näytönohjaimen kanssa kommunikointiin on SPI-väylä (Serial Peripheral Interface). Tämä on kaksisuuntainen ja synkroninen väylä eli kellosignaalin lisäksi siinä on datasignaali molempiin suuntiin [12, s. 1032]. Tavallisen SPI-väylän lisäksi näytöissä on tapana käyttää DC-signaalia (Data / Command), joka määrittelee, lähetetäänkö näytölle ohjaussignaalia vai kuvatietoa. Sähköpaperinäytöissä voidaan myös käyttää Busy-signaalia, joka ilmaisee mikroprosessorille, että laite on tekemässä jotain, mitä ei saa keskeyttää [8]. Lisäksi ohjaimessa on yleensä reset-signaali, jonka avulla laitteen voi alustaa tunnettuun tilaan käynnistyksen jälkeen. [12, s. 840]

Vielä yksi näytönohjaimen toiminnallisuus on tekstin muodostus. Sen avulla näytölle voi lähettää kuvadatan sijaan tekstidataa, jonka perusteella ohjainpiiri muodostaa näytölle tekstiä vastaavat merkit. Tämä kuitenkin koskee lähinnä pieniä näyttöjä, joihin on sisäänrakennettu yksi tai kaksi erikokoista kirjasintyyppiä. [12, s. 840]

Sähköpaperinäyttö vaatii myös sopivan käyttöjännitteen toimiakseen. Tarkemmin ottaen se tarvitsee kaksipuolisen jännitteen, jonka suuruus on noin 20 volttia. Sopivien jännitteiden muodostamiseen on olemassa siihen tarkoitettuja piirejä, mutta joissakin näytöissä tämä hakkuripiiri on integroitu ohjaimeen. Kaikkia hakkurikytkennän vaatimia komponentteja ei kuitenkaan voida integroida näyttöön niiden suuren koon vuoksi, joten ne on lisättävä kytkentään erikseen. [10]

Sähköpaperi on suhteellisen tuore tekniikka, joten sitä käytettäessä on olennaista tarkastella näytön elinikää. Koska näyttö on suurimman osan ajasta lepotilassa, sen käyttöikää rajoittaa ainoastaan päivityskertojen määrä. Esimerkiksi Waveshare Electronicsin 4,2 tuuman näytölle luvataan 1 000 000 päivityskerran elinikää kunhan ympäristön lämpötila on 0 - 40 asteen välillä [8]. Tämä luku tarkoittaisi 15 minuutin päivitysvälillä 27 vuoden käyttöikää, mikä riittää aivan varmasti tähän käyttötarkoitukseen. Lisäksi näyttöä on tarkoitus päivittää mahdollisimman harvoin, joten käyttöikä on todennäköisesti tätäkin pidempi.

2.1.3 WiFi-moduuli

Tässä työssä suunniteltavan laitteen on tarkoitus ottaa yhteys WiFi-verkkoon ja hakea sen kautta tietoa Internet-verkosta. Periaatteessa tähän tarvittaisiin kaksi eri komponenttia. Yksi komponentti hoitaa verkkoyhteydet ja toinen ohjaa ensimmäistä komponenttia sekä näyttöä. Käytännössä on kuitenkin olemassa ratkaisu, jossa sama komponentti kykenee suorittamaan kaikki edellä mainitut tehtävät. Tässä työssä onkin tarkoitus käyttää sellaista SoC-piiriä (System on Chip), joka sisältää WiFi-yhteyden muodostamiseen tarvittavan elektroniikan, prosessoriytimiä, käyttömuistia ja muita oheislohkoja. WiFi-yhteyden muodostamiseen tarvitaan tällaisen piirin lisäksi vain muutamia passiivisia komponentteja, joita ei voida integroida piiriin fyysisen kokonsa vuoksi.

Tässä työssä on alusta asti päätetty käyttää Espressif Systemsin valmistamaa ESP32 nimistä SoC-mikropiiriä. ESP32 on valittu tämän työn laitteen toteutukseen johtuen sen edullisesta hinnasta, avoimesta kehitysympäristöstä sekä avoimesti saatavilla olevan dokumentaation suuresta määrästä. Edullisuudestaan huolimatta sen ominaisuudet ovat riittävät tämän työn toteutukseen. ESP32 sisältää tuen IEEE 802.11 b/g/n -verkkoon, BLE-verkkoon (Bluetooth Low Energy), kaksiytimisen 32-bittisen prosessorin ja 520 kilotavua käyttömuistia. Lisäksi ESP32-mikropiirissä on oleellisia lisälaitteita, kuten pienitehoinen ULP-prosessoriydin (Ultra Low Power), monikanavainen ADC-muunnin (Analog to Digital Converter), SPI-väylä sekä tuki kosketuspainikkeiden toteuttamiseen. Lisäksi piirissä on tuettuna muutamia eri unitiloja, joita tarvitaan vähävirtaisen laitteen tekemiseen. ESP32-piiri ei kuitenkaan sisällä flash-muistia ohjelman tallentamiseen. [13]

Kyseistä mikropiiriä voisi käyttää sellaisenaan, mutta se vaatisi monimutkaista piirilevysuunnittelua johtuen WiFi-verkon korkeista taajuuksista. Tämän takia on tarjolla valmiita moduuleja, jotka sisältävät ESP32-piirin lisäksi muun muuassa WiFi-antennin, flash-muistia, metallisen kannen häiriöiden estämiseksi sekä muutamia passiivisia komponentteja. Vastaavia moduuleja löytyy usean eri valmistajan valikoimasta, myös Espressif Systemsiltä, jonka valmistamaa mikropiiriä työssä käytetään. Esimerkkinä näistä on ESP32-WROOM-moduuli, jossa on kaksiytiminen prosessori, ESP32-SOLO-moduuli, jossa on yksiytiminen prosessori, sekä ESP32-WROVER-moduuli, jossa on kaksiytimisen prosessorin lisäksi integroituna PSRAM-lisämuisti (Pseudo Static Random Access Memory) [14].

Tässä työssä halutaan käyttää kaksiytimistä prosessoria, sillä tämä mahdollistaa WiFi-yhteyttä ylläpitävän ohjelman ja käyttäjän kirjoittaman ohjelman samanaikaisen suorituksen. Tässä työssä ei kuitenkaan tarvita PSRAM-lisämuistia, joten valitaan näistä ESP32-WROOM. WROOM-moduuleistakin on olemassa erilaisia versioita, jotka on esitetty taulukossa 2.3.

Taulukko 2.3. *ESP32 moduulien ominaisuuksien vertailu [14].*

Moduulin malli	Muisti (MB)	ESP32 malli	Antenni
ESP32-WROOM-32	4	D0WDQ6	Sisäinen
ESP32-WROOM-32D	4/8/16	D0WD	Sisäinen
ESP32-WROOM-32SE	4	D0WD	Sisäinen
ESP32-WROOM-32U	4/8/16	D0WD	Ulkoinen
ESP32-WROOM-32E	4/8/16	D0WD-V3	Sisäinen

Taulukossa 2.3 ylimpänä esitetty 32-päätteinen moduuli on ensimmäinen versio WROOM-moduulista ja 32D on siitä paranneltu versio [15]. 32SE-moduuli sisältää erillisen piirin ESP32-piirin salaustoimintojen nopeuttamiseen, mitä tässä työssä ei tarvita. 32U-moduuli ei sisällä muiden tapaan sisäänrakennettua antennia ja sen sijaan moduulissa on antenniliitin, johon voi liittää ulkoisen antennin. Sisäänrakennettu antenni on kuitenkin riittävä tämän työn toteuttamiseen, joten ulkoisen antennin käyttö ei ole tarpeen. 32E-moduuli sisältää uusimman revision ESP32-piiristä, joka korjaa useita edellisten revisioiden ongelmia [16]. Tämä moduuli on kuitenkin niin uusi, ettei sitä ollut saatavilla, kun laitteen prototyyppiä toteutettiin. Tämän perusteella parhaiten työn toteuttamiseen sopii ESP32-WROOM-32D-moduuli. [14]

2.1.4 Energiavarasto

Tässä työssä toteutettavasta laitteesta halutaan täysin langaton, joten siinä on oltava jokin energiavarasto, josta se saa sähkönsä. Yleisimpiä energiavarastoja ovat paristot ja akut. Näistä molemmista on olemassa useita eri tyyppejä ja molemmilla on hyvät ja huonot puolensa. Lisäksi energiavaraston valinta riippuu laitteen muista yksityiskohdista, esimerkiksi luvussa 2.1.5 käsiteltävästä jännitemuunnoksesta sekä lopullisen laitteen

virrankulutuksesta ja halutusta toiminta-ajasta. Akuista tässä tarkastellaan ainoastaan litiumioniakkuja, koska muiden akkuteknologioiden itsepurkautuminen on huomattavasti nopeampaa [17, s. 17-22]. Lisäksi litiumioniakku on muita akkuja kevyempi [17, s. 43].

Sekä paristojen että akkujen tapauksessa energiavaraston fyysinen koko on suoraan verrannollinen sen kapasiteettiin [17, s. 4]. Tämä tarkoittaa sitä, että laitteen virrankulutus, energiavaraston fyysinen koko ja laitteen käyttöaika liittyvät toisiinsa. Laitteen suunnittelun lähtökohtana on järkevää pitää haluttua toiminta-aikaa, koska se vaikuttaa suoraan laitteen käyttömukavuuteen. Toisaalta laitteesta halutaan usein mahdollisimman kompakti eli energiavaraston on oltava tarpeeksi pieni. Päätavoite tällaisen laitteen suunnittelussa onkin tehdä siitä mahdollisimman vähän energiaa kuluttava.

Paristojen suurin etu on niiden edullisuus. Paristokäyttöiseen laitteeseen ei myöskään tarvitse suunnitella latausmahdollisuutta eikä sitä tarvitse siirtää mihinkään ladattavaksi. Paristot eivät yleensä ole kovin litteitä, joten paristokäyttöisestä laitteesta ei yksinkertaisesti saa litteää. Lisäksi laitteen koteloon on suunniteltava luukku paristojen vaihtamiseen.

Akkuja on taas olemassa kovin eri muotoisia ja esimerkiksi tällaiseen melko litteään laitteeseen sopisi hyvin litteä akku. Akkua käytettäessä on kuitenkin suunniteltava sen latausmekanismi, joka voi olla laitteeseen integroitu latauspiiri, jolloin laitteen lataaminen tapahtuisi esimerkiksi yleisesti käytetyn micro-usb-liittimen avulla [17, s. 63]. Tällöin laite on kuitenkin joko vietävä lataukseen tai tuotava laturi laitteen viereen. Ensimmäisessä tapauksessa laite ei ole käytettävissä halutussa paikassa latauksen ajan. Toisessa tapauksessa on mahdollista, että käyttäjä jättää laitteen jatkuvasti lataukseen etenkin, jos akkukesto on liian lyhyt. Silloin laitteen sisältämä akku on melko turha.

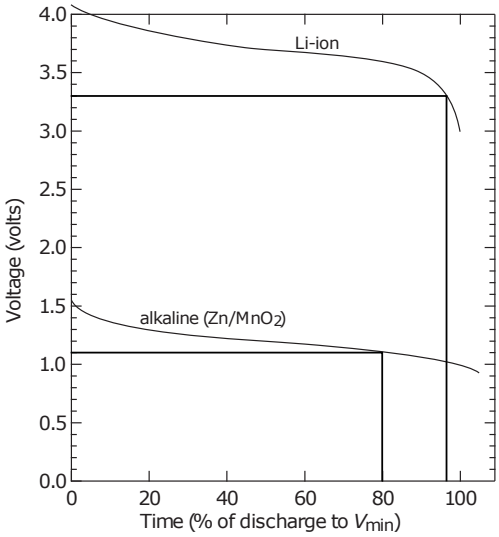
Paristojen sekä akun jännite muuttuu varauksen mukaan ja molemmilla on omat jännitealueensa. Yhden alkaliparistoon napajännite on täytenä 1,61 - 1,73 volttia ja tyhjäksi se luokitellaan, kun napajännite on 0,8 - 1,0 volttia [18, s. 9.15]. Litiumakun jännitteen turvallinen jännitealue sen sijaan on 2,5 - 4,3 volttia [18, s. 26.2]. Paristojen jännitealue voidaan saada akkua vastaavaksi, jos niitä kytketään kolme kappaletta sarjaan. Tällöin paristojen jännitteet summautuvat ja jännitealueeksi saadaan 2,4 - 5,2 volttia, mikä on tarpeeksi lähellä akun jännitettä. Tämän seurauksena laitteesta voidaan tehdä sellainen, että se voi toimia sekä kolmella alkaliparistolla että yhdellä litiumioniakulla.

2.1.5 Jännitemuunnos

Elektroniset komponentit vaativat tietyn käyttöjännitteen toimiakseen. Liian suuri jännite rikkoo komponentin peruuttamattomasti ja liian pieni jännite ei riitä komponentin oikeaan toimintaan ja voi aiheuttaa virheellistä toimintaa. Eräs yleinen käyttöjännite on 3,3 volttia ja sen tyypillinen toleranssi on $\pm 10\,\%$, mikä tarkoittaisi 3,0 - 3,6 voltin jännitettä [19]. Tässä työssä käytettävän ESP32-piirin käyttöjännitealue on 2,3 - 3,6 volttia, mutta ESP32-WROOM-32D moduulin käyttöjännite on edellä mainittu 3,0 - 3,6 volttia, mahdollisesti johtuen sen sisältämästä flash-piiristä [13][20].

Luvussa 2.1.4 mainitut paristojen ja akun jännitteet eivät kuitenkaan täsmää työssä käytettävien komponenttien käyttöjännitteisiin, joten on tehtävä jännitemuunnos. Jännitemuunnoksen tekemiseen on olemassa kaksi vaihtoehtoa: lineaarinen jänniteregulaattori ja hakkuriregulaattori [12, s. 594]. Molempien tapauksessa on kuitenkin huomioitava regulaattorin virrankesto, ulostulojännite ja tyhjäkäyntivirta. Regulaattorin virrankestolla tarkoitetaan sitä ulostulovirtaa, jota se on tarkoitettu kestämään. Ulostulojännite sen sijaan tarkoittaa sitä jännitettä, joka halutaan regulaattorin ulostuloon. Tyhjäkäyntivirta taas tarkoittaa itse regulaattorin virrankulutusta eli virrankulutusta siinä tapauksessa, kun regulaattorin ulostulovirta on nolla. Vaadittavan virrankeston ja ulostulojännitteen määrittävät kytkennässä käytettävät komponentit. Tyhjäkäyntivirralle ei sen sijaan ole erityistä vaatimusta, mutta se halutaan minimoida, jotta paristokäyttöisen laitteen käyttöaika on mahdollisimman pitkä. [12]

Lineaarinen jänniteregulaattori perustuu puhtaasti häviölliseen toimintaan. Se sisältää automaattisesti kuormitusvirran mukaan ohjautuvan komponentin, joka aiheuttaa jännitehäviön sisäänmenon ja ulostulon väliin niin, että ulostulojännite on tietyn arvoinen. Edellä mainittu jännitehäviö muuttuu lineaariregulaattorissa lämmöksi. Lineaariregulaattorin etu on sen edullinen hinta ja tarvittavien oheiskomponenttien pieni määrä. Suurin lineaariregulaattorin heikkous on sen hyötysuhde, joka saadaan yhtälöstä $\eta = V_{out}/V_{in}$. Esimerkiksi kolmen uuden alkaliparistoon ja 3,3 voltin ulostulon kanssa tämä tarkoittaisi 67% hyötysuhdetta. Lisäksi lineaariregulaattorin toimintaperiaatteesta johtuen sen ulostulojännite ei voi koskaan olla isompi kuin sisäänmenojännite. Tämä taas tarkoittaisi sitä, että lineaariregulaattoria käytettäessä laite lakkaa toimimasta jo ennen paristojen tai akun tyhjentymistä, jos ulostulona tarvitaan 3,3 volttia. Lineaarisen regulaattorin tapauksessa on myös otettava huomioon sen pienin mahdollinen jännitehäviö, joka vaihtelee suuresti eri lineaariregulaattoreilla. [12, s. 600]



Kuva 2.3. *Alkaliparistoon ja litiumioniakun jännitekäyrät [12].*

Hakkuriregulaattori taas pyrkii reguloimaan ulostulojännitettä häviöttömästi ja perustuu kelan kykyyn varastoida energiaa. Hakkuriregulaattorissa kelaan kytketään virta vuoro-

tellen päälle ja pois käyttäen puolijohdekytkintä. Kytkimen ollessa kiinni kelan virta kasvaa ja täten myös siihen varastoitunut energia. Kytkimen ollessa auki kelaan varastoitunut energia siirtyy ulostulokondensaattoriin, joka tasoittaa ulostulon jännitevaihtelut. Hakkuriregulaattorissa täytyy myös olla jokin komponentti, joka ohjaa puolijohdekytkintä ulostulojännitteen perusteella. Tarkemmin ottaen hakkuriregulaattori ohjaa puolijohdekytkimen kiinni- ja aukiolon pulssisuhdetta. Hakkuriregulaattorin suurin etu on sen hyötysuhde, joka on usein yli 90% riippumatta sisäänmenojännitteestä. Lisäksi hakkuriregulaattori voi toimia sekä ulostulojännitettä pienemmällä että suuremmalla sisäänmenojännitteellä. Hakkurikytkennän heikkoutena on vastaavasti sen hinta ja tarvittavien komponenttien määrä. [12, s. 636]

Regulaattorin valintaan vaikuttaa myös energiavaraston valinta. Kuten edellä käsiteltiin, lineaariregulaattori ei voi toimia haluttua ulostuloa pienemmällä sisäänmenojännitteellä. Jos lineaariregulaattoriin kytketään kolme alkaliparistoa, kytkentä lakkaa toimimasta, kun yksittäisen pariston jännite on alle 1,1 volttia. Kuvasta 2.3 voidaan nähdä, että alkalipariston tapauksessa jää käyttämättä noin 20% varauksesta. Ratkaisuna voidaan kytkeä useamman alkalipariston sarjaan, mutta silloin lineaariregulaattorin hyötysuhde olisi vielä huonompi. Litiumioniakun tapauksessa tällaista ongelmaa ei ole, vaan suurin osa varauksesta saadaan käytettyä myös lineaariregulaattoria käytettäessä.

2.1.6 Oheiskytkennät

Kuten edellä on käsitelty, mikropiirien kannalta on oleellista, että niiden käyttöjännite on tiettyjen rajojen sisällä ja tästä huolehtii jänniteregulaattori. Jänniteregulaattori ei kuitenkaan voi luoda energiaa tyhjästä, jos esimerkiksi paristo tai akku on lähes tyhjä. Tämä voi johtaa tilanteeseen, jossa mikropiiriin käyttöjännite on sen sallitun käyttöjännitealueen alarajan alapuolella. Tällainen tilanne taas voi johtaa mikropiirin virheelliseen toimintaan. Ratkaisuna tähän ongelmaan on valvontapiiri, jonka tehtävänä on tarkkailla käyttöjännitettä [12, s. 755]. Käyttöjännitteen laskiessa tietyn rajan alapuolelle valvontapiiri estää siihen kytkettyjä komponentteja toimimasta, kunnes jännite on noussut tarpeeksi ja alarajan ylityksestä on kulunut tietty aika.

Lisäksi kytkentään tarvitaan käyttöliittymä, joka mahdollistaa tiedon esittämisen ja tiedon syöttämisen. Tiedon esittäminen tapahtuu pääasiassa näytön avulla. Tämän lisäksi on kuitenkin hyvä olla jokin merkkivalo, joka esittää esimerkiksi sitä, että laite on hakemassa tietoa Internet-verkosta. Merkkivalon avulla voi myös ilmaista, että laite on sellaisessa tilassa, jossa laitteen asetuksia voi vaihtaa. Merkkivalo on myös hyvä apu kehitysvaiheessa. Merkkivaloja voi olla laitteessa useampi tai yksi merkkivalo voi ilmaista laitteen eri tiloja esimerkiksi vilkkumalla eri tahtiin. Merkkivalon sijaan kaiken tarvittavan voisi esittää myös näytöllä, mutta näytön päivittäminen kuluttaisi huomattavasti enemmän virtaa kuin pieni merkkivalo.

Koska laitteen on tarkoitus olla pääasiassa itsenäisesti toimiva infonäyttö, ei siinä tarvita hienostuneita menetelmiä tiedon syöttämiseen. Laitteessa täytyy kuitenkin olla jokin tapa

asettaa se sellaiseen tilaan, jossa sen asetuksien vaihtaminen on mahdollista. Lisäksi laitteessa voisi olla mahdollisuus päivittää näyttö halutulla hetkellä. Molempiin tarkoituksiin riittää painonappi tai kosketusnappi. Asetuksien vaihtamista varten laitteessa voisi olla näppäimistö, mutta se ei ole mielekästä, koska laitteen on tarkoitus olla kompakti ja asetuksia ei ole tarkoitus vaihtaa usein.

Mikäli laitteesta halutaan akkukäyttöinen, on laitteessa oltava mahdollisuus akun lataamiseen. Tämä onnistuu kätevästi, kunhan laitteessa on jokin latausliitin sekä akun lataamiseen tarkoitettu mikropiiri. Tämän lisäksi tarvitaan pieni määrä passiivisia komponentteja ja haluttaessa latausta ilmaiseva merkkivalo.

Akku- tai paristokäyttöisyyden takia laitteen on myös kyettävä mittaamaan akun tai pariston käyttöjännite. Molempien tapauksessa laite voisi arvioida jäljellä olevan varaustason ja ilmoittaa sen näytöllä. Lisäksi laite voisi ilmoittaa paristojen vaihtamisen tai akun latauksen tarpeesta ja olla päivittämättä näyttöä kunnes käyttäjä vaihtaa paristot tai laittaa laitteen lataukseen. Tämä varmistaisi sen, ettei edellä mainittu kehotus pyyhkiydy näytöltä, jos laite alkaa toimia väärin käyttöjännitteen laskemisen myötä. Akkukäyttöisen laitteen tapauksessa jännitteen mittaus on erityisen tärkeää, sillä litiumioniakku voi vaurioitua, mikäli sen jännite laskee turvallisen alarajan alapuolelle [17, s. 98]. Mikäli jännite on liian alhainen, on varmistettava, ettei laite tyhjennä akkua yhtään enempää. Tämä onnistuu esimerkiksi niin, että ESP32-piiri siirtyy unitilaan, eikä herää ennen kuin laite kytketään lataukseen.

Laitteen kehittämisen kannalta laitteessa on myös hyvä olla muun muuassa ohjelmointiliitin, jonka kautta laitteen voi ohjelmoida sopivalla ohjelmointilaitteella. Lisäksi laitteessa kannattaa olla testipisteitä (testpoint) [21, s. 20.27], joiden avulla laitteeseen saa kehitysvaiheessa tarvittaessa kytkettyä sopivilla testauskoukuilla teholähteitä ja mittalaitteita. Edellä mainittu merkkivalo helpottaa myös laitteen kehittämistä, koska se voi ilmoittaa, missä tilassa laite on.

2.2 Ohjelmisto

Tässä työssä suunniteltavassa laitteessa on tarkoitus käyttää mahdollisimman paljon ohjelmoitavan piirin valmistajan tarjoamia kirjastoja. Tässä tapauksessa kyseessä on Espressif Systemsin tarjoama ESP-IDF-paketti [22]. Tässä paketissa on mukana muun muassa reaaliaikainen käyttöjärjestelmä (FreeRTOS) ja tarvittavat kirjastot WiFi-yhteyden ja Internet-yhteyksien muodostamiseen. Reaaliaikainen käyttöjärjestelmä mahdollistaa tehtävien (task) rekisteröimisen ja ajoittamisen niin, että käyttöjärjestelmä huolehtii niiden suorittamisesta [23, s. 159]. WiFi- ja Internet-kirjastot taas mahdollistavat yhteyksien avaamisen ja ohjaamisen yksinkertaisten funktiokutsujen avulla.

2.2.1 Langaton lähiverkko

Tässä työssä toteutettavan laitteen on tarkoitus käyttää langatonta lähiverkkoa. Tämä mahdollistaa sen, että laite on täysin langaton. Yleisin langattomista lähiverkoista on

IEEE 802.11 -standardiin perustuva lähiverkko, jota kutsutaan useimmiten markkinointitermillä WiFi. WiFi-verkkojen käyttö on yleistynyt viime aikoina muun muuassa koteihin tarkoitettujen langattomien Internet-liittymien vuoksi, sillä niiden käyttöön tarkoitetussa modeemissa on usein WiFi-yhteys eikä lainkaan mahdollisuutta langallisen lähiverkon muodostamiseen. [24]

WiFi-verkko toimii aina tietyllä taajuudella, jota kutsutaan myös kanavaksi. IEEE 802.11b -standardi määrittelee 14 eri kanavaa, joista Euroopassa on käytettävissä vain 13. Tukiasemaan yhdistettäessä päätelaitteen täytyy käydä nämä kanavat läpi ja selvittää, mitä kanavaa haluttu tukiasema käyttää. [24]

WiFi-verkkoa käytettäessä on otettava huomioon sen turvallisuus. Pahaa aikovan ihmisen on helpompi salakuunnella tai yksinkertaisesti käyttää langatonta verkkoa, kuin langallista verkkoa. Tämä johtuu siitä, että langattoman verkon käyttämiseen riittää, että on kyseisen verkon kuuluvuusalueella. Tämän takia WiFi-verkkoihin on toteutettu erilaisia tunnistautumis- ja salausmekanismeja. Käytetyin näistä on WPA2-protokolla (Wi-Fi Protected Access) ja tarkemmin WPA-Personal, joskin yrityksissä käytetään usein WPA-Enterprise-versiota. [25]

ESP32-piiri tukee WiFi-verkkojen kannalta useampaa eri moodia. Se voi toimia tukiasemana (Access Point), päätelaitteena (Station) tai näiden yhdistelmänä. Yhdistelmä tarkoittaa sitä, että ESP32 voi olla yhteydessä toiseen tukiasemaan päätelaitteena ja toimia tukiasemana muille päätelaitteille yhtä aikaa. ESP32-piiri tukee myös edellä mainittuja WPA-Personal ja WPA-Enterprise-salauksia. [15]

Tässä laitteessa oleellisin on toiminta päätelaitteena. Ohjelman on ensin ladattava WiFi-yhteyden muodostamiseen tarvittavat tiedot laitteen flash-muistista. Verkkoon yhdistämisen kannalta oleelliset tiedot ovat verkon SSID-tunniste (Service Set Identifier) sekä salasana. Tämän jälkeen WiFi-yhteyden voi muodostaa. WiFi-yhteyden mudostamisessa voi kulua useita sekunteja, mutta ohjelman ei tarvitse odottaa sitä, koska ESP-IDF sisältää mekanismin, jolla voi automaattisesti käynnistää ohjelman seuraavan vaiheen.

2.2.2 Kellonaika

Tässä työssä toteutettavan laitteen on tarkoitus hakea palveluntarjoajan palvelimelta tulevat ja menossa olevat tapahtumat, joten sen on tiedettävä kellonaika tapahtumien hakuhetkellä. Toteutettavassa laitteessa pitäisi olla tätä varten kellonaikaa ylläpitävä ominaisuus sekä mahdollisuus kellonajan syöttämiseen. Tämä ei kuitenkaan ole välttämätöntä, sillä laite on kytketty verkkoon ja se voi hakea kellonajan itse. Kellonajan voi hakea esimerkiksi siihen tarkoitetulta SNTP-palvelimelta (Simple Network Time Protocol). [26]

Toinen vaihtoehto kellonajan selvittämiseen on käyttää kalenteripalvelimen lähettämää HTTP-vastausta (Hyper Text Transfer Protocol). HTTP-palvelimen vastauksessa on oltava vastauksen lähetysaika ja päivämäärä RFC 822 -standardin (Request For Comments) määrittelemässä muodossa, mikäli vastaus on muuta kuin uudelleenohjaus tai virheilmoitus

[27]. Mikäli kalenteritietojen haku onnistuu, voidaan olettaa, että kellonajan selvittäminen onnistuu palvelimen lähettämästä HTTP-vastauksesta. Näin ollen kellonaikaa ei tarvitse hakea erikseen SNTP-palvelimelta, mikä yksinkertaistaa laitteen toimintaa ja mahdollisesti pienentää virrankulutusta. [28]

Kellonaikaa käsitellessä täytyy ottaa huomioon muun muuassa aikavyöhyke ja kesäaika. SNTP- ja HTTP-palvelimien vastauksissa, kuten myös kalenterin aikaleimoissa, ei ole välttämättä käytetty samaa aikavyöhykettä kuin millä laitetta käytetään. Tämä tarkoittaa sitä, että laitteeseen on oltava mahdollista asettaa aikavyöhyke sekä tieto siitä, onko kesäaika käytössä. Tämän jälkeen laite voi muuttaa verkosta haettavat kellonajat käytössä olevan aikavyöhykkeen mukaisiksi. Aikavyöhykkeen vuoksi kellonaikaa korjattaessa on myös mahdollista, että tapahtuma siirtyy päivästä toiseen. Tällöin on huomioitava myös kuukauden vaihtuminen sekä eri kuukausien pituudet. Helmikuun tapauksessa on erityisesti otettava huomioon karkausvuosi, jolloin kuukaudessa on poikkeuksellisesti 29 päivää.

2.2.3 Kalenteritiedon haku

Nykyaikainen Internet-verkko perustuu niin sanottuun TCP/IP-yhdistelmään, jossa TCP-yhteys (Transmission Control Protocol) muodostaa kuljetuskerroksen ja IP-yhteys (Internet Protocol) muodostaa verkkokerroksen. Tästä seuraavana voi olla esimerkiksi HTTP-yhteys, joka muodostaa sovelluskerroksen. Käytännössä tämä tarkoittaa sitä, että verkossa oleva laite muodostaa yhteyden toiseen verkossa olevaan laitteeseen IP-osoitteen ja TCP-portin perusteella. IP-osoitteet ovat verkossa palveluntarjoajan määrittelemiä ja lähiverkossa reitittimen määrittelemiä. TCP-portti sen sijaan on yleensä sidottu tiettyyn sovellukseen, esimerkiksi HTTP-yhteyden portti on joko 80 tai 8080. Internet-sivuja selatessa käyttäjän ei kuitenkaan tarvitse muistaa numeromuotoisia IP-osoitteita, vaan tekstimuotoinen osoite riittää. Tätä tekstimuotoista osoitetta vastaavan numeromuotoisen IP-osoitteen voi selvittää DNS-palvelun (Domain Name System) avulla. [29]

Normaalisti HTTP-yhteyden informaatio siirtyy portissa 80 salaamattomana. Tämä tarkoittaa sitä, että kuka tahansa samassa verkossa oleva voi vakoilla liikennettä palvelimen ja käyttäjän välillä. Kalenterin tapauksessa tämä tarkoittaisi sitä, että ulkopuolinen voisi saada käsiinsä käyttäjän kalenterin osoitteen, kalenterissa olevat tapahtumat tai nämä molemmat. Tämän takia on kehitetty erinäisiä salaustekniikoita joista yleisimmät ovat SSL (Secure Sockets Layer) sekä siitä kehitetty TLS (Transport Layer Security). Nämä voivat mahdollistaa sekä yhteyden osapuolten identiteetin varmistamisen että yhteyden salauksen. Osapuolten identiteetin tarkistamista ei käsitellä tässä työssä. [30]

Käytännössä TLS-salaus toimii niin, että ensimmäisen yhteydenoton jälkeen palvelin lähettää laitteelle oman sertifikaattinsa, joka sisältää muun muuassa palvelimen julkisen salausavaimen. Tämän jälkeen laite vastaa palvelimelle edellä mainitulla julkisella avaimella salatulla viestillä, jonka salaus palvelimen on purettava oman salaisen avaimensa avulla. Tämän jälkeen molemmat osapuolet muodostavat yhteyskohtaisen salausavaimen käyttäen muun muuassa edellä mainittua viestiä, jonka sisällön tuntevat vain sen salaava

laite sekä palvelin, jolla on oikea salainen avain viestin purkamiseen. Mikäli molemmat osapuolet muodostavat saman salausavaimen, on sertifikaatin oikeellisuus taattu ja yhteyden osapuolet voivat jatkaa salattua kommunikaatiota käyttäen generoituja salausavaimia. [31]

Kalenteritiedon haun työssä käytettävästä Google Calendar -palvelusta voi toteuttaa usealla eri tavalla. Ensimmäinen vaihtoehto on Googlen virallisen Calendar API -rajapinnan (Application Programming Interface) käyttö [32]. Kyseisen rajapinnan kautta voi esimerkiksi luetella, lisätä ja poistaa tapahtumia julkisista ja yksityisistä kalentereista. Rajapinnan käyttöön tarvitaan avain ja kullekin avaimelle pitää sallia pääsy yksityisiin kalentereihin. Rajapinnan etuna on myös mahdollisuus hakea kalenteritapahtumat tietyltä aikaväliltä, esimerkiksi kaikki tulevat tapahtumat. Tämän rajapinnan käyttö olisi ehkä helpoin ja turvallisin tapa hakea kalenteritiedot, mutta se on palveluntarjoajakohtainen toteutus eikä näin ollen toimi muiden kalenterien kanssa.

Toinen vaihtoehto on oman skriptin kirjoittaminen Google Apps Script -palvelussa [33]. Tämä tarkoittaa käytännössä oman rajapinnan toteuttamista virallisen sijaan. Tämä mahdollistaa esimerkiksi sellaisen rajapinnan toteutuksen, joka ei vaadi todennusta ja mahdollistaa ainoastaan kalenteritietojen lukemisen. Tämä olisi vielä helpompaa kuin ensimmäinen vaihtoehto, mutta on samasta syystä palveluntarjoajakohtainen eikä täten itsenäisesti toimiva.

Kolmas vaihtoehto on hakea kalenteritiedot kalenterin julkisen linkin kautta iCalendar-muodossa. Tämä on perinteinen kalenteritietojen jakamiseen tarkoitettu menetelmä, joka mahdollistaa vain kalenteritietojen luvun. iCalendar-formaatti on laajasti tuettu ja on täten yleisesti pätevä ratkaisu ja toimisi esimerkiksi WebCal.fi sivustolle [34]. Sen haittapuolena on todennuksen puute ja se, että kalenteritietoja hakiessa on pakko ladata kaikki kalenterissa olevat tapahtumat. Edellä mainittujen rajapintojen tapauksessa olisi mahdollista hakea vain halutun aikavälin kalenteritapahtumat. Edellä käsiteltyjen menelmien keskeisimmät ominaisuudet on esitetty tiivistetysti taulukossa 2.4. [35]

Taulukko 2.4. *Kalenteritiedon hakuvaihtoehdot [32][33][36].*

Toteutus	Itsenäinen toiminta	Todennus
Google Calendar API	Ei	Kyllä
Google Apps Script	Ei	Kyllä/Ei
iCalendar linkki	Kyllä	Ei

Kalenteritiedon haku edellyttää Internet-yhteyden muodostamista kalenterisivuston palvelimeen. Esimerkkinä käytettävä Google Calendar vaatii TLS-salauksen käyttämistä [37], mikä on otettava huomioon ohjelmaa kirjoitettaessa. On avattava yhteys palvelimen osoitteeseen tiettyyn porttiin, joka on HTTP-yhteyden määrittelemä eli 80. Salauksen käyttö voi aiheuttaa uudelleenohjauksen TLS-yhteyksissä käytettyyn porttiin, joka on yleensä 443 [38]. Tämän jälkeen palvelimelle on lähetettävä HTTP-kysely, johon palvelin vastaa HTTP-vastauksella, sekä haettavalla datalla. Tätä voi testata curl-sovelluksella verbose-vivun

kanssa, jolloin nähdään HTTP-kysely sekä vastaus [39]. Tämä on esitetty listauksessa 2.1, josta on kuitenkin jätetty pois epäolennainen informaatio. Kyseisestä listauksesta nähdään päätelaitteen GET-kysely, palvelimen HTTP/2 200 -vastaus sekä tyhjä iCalendar-kalenteri. Jos kysely on viallinen, vastaa palvelin, ettei kyselyssä pyydettävää dataa voida hakea. Vastauksen saamisen jälkeen yhteys on suljettava.

```
> GET /calendar/ical/.../basic.ics HTTP/2
> Host: calendar.google.com
>
...
< HTTP/2 200
< content-type: text/calendar; charset=UTF-8
< date: Thu, 27 Feb 2020 16:50:44 GMT
<
BEGIN:VCALENDAR
PRODID:-//Google Inc//Google Calendar 70.9054//EN
VERSION:2.0
CALSCALE:GREGORIAN
METHOD:PUBLISH
X-WR-CALNAME:Testikalenteri
X-WR-TIMEZONE:Europe/Helsinki
END:VCALENDAR
```

Listaus 2.1. *curl-ohjelman tuloste kalenteritietojen haussa.*

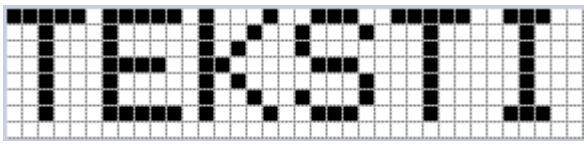
Kalenteripalvelimen lähettämä tieto on iCalendar-muodossa ja on helposti koneluettavassa muodossa. Vastaanotettua tietoa pitää kuitenkin suodattaa ja käsitellä sopivampaan muotoon. Kalenteritiedosto voi olla hyvinkin iso riippuen sen sisältämien tapahtumien määrästä, eikä rakennettavassa laitteessa voida olettaa, että koko tiedosto mahtuu kerralla muistiin. Tämän takia kalenteritietoja pitää käsitellä sitä mukaa, kun palvelin lähettää niitä, eli algoritmin on oltava sellainen, että dataa voi käsitellä sopivissa osissa.

iCalendar-formaatin rakenne on hyvin yksinkertainen. Se nojaa vahvasti rivin alussa oleviin avainsanoihin, joiden jälkeen tulee avainsanaa vastaava informaatio kaksoispisteellä erotettuna. Avainsana BEGIN tarkoittaa jonkin lohkon, esimerkiksi kalenterin (VCALENDAR) tai tapahtuman (VEVENT), alkua. Avainsana END tarkoittaa edellä mainittujen lohkojen loppumista. Lisäksi oleellisia ovat kalenteritapahtuman kuvaus (SUMMARY), alkuajankohta (DTSTART), loppumisajankohta (DTEND) sekä sijainti (LOCATION). Tämän lisäksi oleellinen avainsana on tapahtumien toistoa kuvaava sääntö (RRULE). Poikkeuksena tästä rakenteesta ovat ylipitkät tietokentät, jotka jatkuvat tietyn merkkimäärän jälkeen seuraavalle riville ilman rivin alussa olevaa avainsanaa. [35]

2.2.4 Näytön päivittäminen

Kalenteritietojen hakemisen jälkeen ne on jotenkin jäsennettävä ja esitettävä näytöllä. Näytön koko on rajallinen, joten ensin on valittava, mitä tietoja halutaan näyttää. Tämän jälkeen on päätettävä esitysmuoto. Jos esimerkiksi halutaan pelkästään tapahtuma-listaus, voidaan siinä esittää tapahtuman kuvaus, sen ajankohta sekä sijainti. Jos taas halutaan esittää vain yhden viikon tapahtumat kerrallaan, voidaan esittää pelkästään tapahtuman kuvaus ja mahdollinen kellonaika kunkin viikonpäivän kohdalla. Molempien tapauksessa kalenteritapahtumat on ensin järjestettävä aikajärjestykseen, sillä Google Calendar -palvelu lähettää tapahtumat siinä järjestyksessä, missä ne on lisätty kalenteriin.

Seuraavaksi edellä muodostettu teksti on saatava näytölle. Kuten luvussa 2.1.2 on käsitelty, on joissakin näytöissä ohjaimeen sisäänrakennettu kirjasintyyppi. Siinä tapauksessa näytön konfiguroinnin jälkeen riittää lähettää sille tekstimuotoista sisältöä, jonka se osaa piirtää itse. Käytetyssä sähköpaperinäytössä ei kuitenkaan sellaista ominaisuutta ole, joten teksti on muodostettava itse. Tätä varten on ensin ladattava laitteeseen kutakin kirjainmerkkiä vastaava kuva. Tämän jälkeen laite voi välittää näytölle merkkien sijaan niitä vastaavat kuvat. Esimerkki tällaisesta 5x8 pisteen kirjasintyypistä on esitetty kuvassa 2.4, jossa alin rivi on tyypillisesti varattuna alleviivausta varten. Mikäli käytössä on näyttö, joka kykenee esittämään useampaa kuin kahta väriä, on tässä vaiheessa myös päätettävä, mitä esitetään milläkin värillä. Jos käytössä on näyttö, joka voi esittää punaista väriä, voi sillä korostaa esimerkiksi meneillään olevia tapahtumia.



Kuva 2.4. *Esimerkki 8x5 kokoisesta, tasalevyisestä kirjasintyypistä.*

Tekstiä käsitellessä on oleellista ottaa huomioon merkkikoodaus. iCalendar-standardi määrittelee, että kalenteritietojen oletusmerkkikoodaus on UTF-8 (Unicode Transformation Format). Listauksessa 2.1 esitetystä HTTP-vastauksesta nähdään, että Google Calendar käyttää myös UTF-8-merkkikoodausta. Laitteseen ei kuitenkaan ole mielekästä tallentaa kaikkia merkkejä, jotka UTF-8-merkkikoodauksella voidaan esittää. Laitteesta täytyy silti löytyä käytössä olevien merkkien kirjasintyyppi. Suomessa tämä tarkoittaa latinalaisten aakkosten lisäksi ääkkösiä. Lisäksi iCalendar-standardi määrittelee erikoismerkit, joiden edessä on oltava kenoviiva. Tällaisia erikoismerkkejä ovat kenoviiva, pilkku ja puolipiste. Tekstiä esitettäessä kenoviiva on poistettava näiden erikoismerkkien edestä. [35]

Kuvatiedon siirtäminen näytölle tapahtuu SPI-väylän kautta yksi tavu kerrallaan. Tätä voi nopeuttaa käyttämällä ESP32-piirin DMA-lohkoa (Direct Memory Access). DMA-lohkolle välitetään osoitin muistialueeseen, minkä jälkeen se huolehtii tietojen siirtämisestä muistista SPI-lohkolle. DMA-lohkon käyttö mahdollistaa ohjelman suorituksen jatkumisen samaan aikaan, kun kuvatiedot siirtyvät näytölle. Näytön päivittämisen jälkeen se on sammutettava virrankulutuksen pienentämiseksi. [12, s. 1010-1011]

2.2.5 Unitila

Tässä työssä toteutettavan laitteen on tarkoitus toimia mahdollisimman pitkään paristojen tai akun varassa. Tämä tarkoittaa sitä, että laitteessa on oltava mahdollisimman pieni virrankulutus. Toteutattavan laitteen on tarkoitus päivittää näyttöä tietyin väliajoin, mutta muulloin sen ei tarvitse olla aktiivisessa tilassa. Mikroprosessoreissa onkin tätä tarkoitusta varten usein toteutettuna jonkinlainen virransäästötila eli unitila. Unitiloja on myös olemassa erilaisia, riippuen siitä mitä piirin lohkoja halutaan jättää päälle. Vastaavasti eri unitiloista herääminen vaatii eri toiminnot. Joistakin unitiloista piiri voi herätä päällä olevan ajastimen avulla ja joissakin unitiloissa tarvitaan ulkoinen heräte, eli esimerkiksi napin painallus.

Esimerkiksi luvussa 2.1.3 mainitulla ESP23-piirillä on karkeasti neljä eri unitilaa ja ne on esitetty taulukossa 2.5. Modem-sleep-tilassa ESP32-piirin prosessori on päällä, mutta WiFi-radio on sammutettuna. Tämä unitila sammuttaa WiFi-radion tietyin väliajoin, kun laite on yhteydessä tukiasemaan, mutta ei lähetä eikä vastaanota mitään. Light-sleep-tilassa ESP32-piirin prosessori on sammutettuna, mutta piiri on sellaisessa tilassa, josta se voi herätä nopeasti. Deep-sleep-tilassa prosessori on kokonaan sammutettuna, ja päällä on vain erillinen RTC-ajastin (Real Time Clock) sekä tähän ajastimeen liittyvä muisti. Prosessorin muisti tyhjenee Deep-sleep-tilassa, joten tästä unitilasta herääminen vastaa ESP32-piirin ensimmäistä käynnistystä. Tästä unitilasta herääminen on hidasta, mutta edellä mainitussa RTC-lohkon muistissa voi säilyttää jonkin verran informaatiota. Viimeisenä on hibernation-unitila, jossa vain edellä mainittu RTC-ajastin on päällä. Laitteessa ei tällöin ole päällä mitään käyttömuistia, jossa voisi säilyttää informaatiota siitä, mitä prosessori teki ennen unitilaan siirtymistä. [13]

Taulukko 2.5. *ESP32-piirin unitilat [13].*

Tila	Virrankulutus
Active	95 - 240 mA
Modem-sleep	20 - 31 mA
Light-sleep	0,8 mA
Deep-sleep	10 uA
Hibernation	5 uA

Laitteessa on kuitenkin flash-muisti, joka ei tarvitse sähköä informaation säilyttämiseen. Flash-muistia voi siis käyttää tiedon pitkäaikaissäilytykseen. Flash-muistin kukin muistisolu kestää kuitenkin rajallisen määrän kirjoituskertoja, joiden jälkeen se lakkaa toimimasta [12, s. 1024]. Lisäksi flash-muistiin kirjoittaminen on suhteellisen hidasta ja kuluttaa paljon enemmän energiaa kuin RAM-muistin (Random Access Memory) ylläpitäminen. Näin ollen harvoin muuttuva informaatio kannattaa tallentaa flash-muistiin ja usein vaihtuva informaatio RTC-lohkon RAM-muistiin, mikäli tietoa halutaan säilyttää unitilan yli. On myös otettava huomioon, että kaikki RAM-muistit tyhjenevät, mikäli laitteesta irrotetaan energialähde. [13]

2.2.6 Asetukset

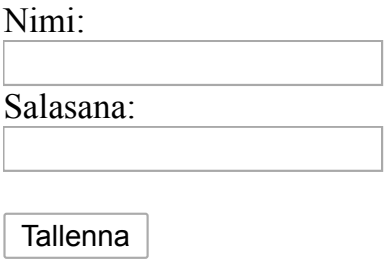
Tässä työssä toteutettavassa laitteessa on oltava jokin mekanismi esimerkiksi WiFi-asetusten ja kalenteriosoitteiden syöttämiseen. Tähän tarkoitukseen laitteessa voisi olla näppäimistö. Kuten luvussa 2.1.6 on käsitelty, useimmiten riittää asettaa laitteen asetukset kerran, kun laite otetaan käyttöön. Tämän takia näppäimistön sisällyttäminen laitteeseen ei ole mielekästä. Vaihtoehtoinen menetelmä on käyttää jotain muuta laitetta, esimerkiksi puhelinta tai tietokonetta, asetusten syöttämiseen. Laitteessa täytyy kuitenkin olla jokin mekanismi laitteen herättämiseen unitilasta ja asetustilaan siirtymiseen, esimerkiksi painonappi.

Kuten luvussa 2.1.3 on kerrottu, on ESP32-piirissä myös tukiasematoiminto. Tämä tarkoittaa sitä, että asetusten syöttämistä varten ESP32-piiri voi toimia WiFi-tukiasemana. Tässä tapauksessa tukiasemaan voi muodostaa yhteyden esimerkiksi puhelimella tai tietokoneella, jossa on WiFi-radio. WiFi-yhteyden muodostamisen jälkeen ESP32-piiri toimisi HTTP-palvelimena ja puhelin tai tietokone toimisi päätelaitteena. Tietyn osoitteen avaaminen päätelaitteessa näyttäisi tällöin palvelimen esittämän asetussivun. Tämän jälkeen laitteen asetuksia voisi säätää kätevästi puhelimesta tai tietokoneesta ja tallentaa ne laitteen muistiin.

```
1 <form action="/action_page.php" method="post">
2   <label for="ssid">Nimi:</label><br>
3   <input type="text" id="ssid" name="ssid"><br>
4   <label for="pwd">Salasana:</label><br>
5   <input type="password" id="pwd" name="pwd"><br><br>
6   <input type="submit" value="Tallenna">
7 </form>
```

Listaus 2.2. *Form-avainsanan avulla luotu HTML-lomake.*

Edellä mainitun palvelimen asetussivun voi tehdä helposti HTML-kielen form-avainsanan avulla. Form-alueen sisällä voidaan määritellä text-tyyppisiä tekstikenttiä, esimerkiksi WiFi-verkon nimeä ja salasanaa varten. Lisäksi tarvitaan submit-tyyppinen nappi asetuksien tallentamista varten. Lisäksi sivulla on hyvä olla tekstikenttiä kuvaavat tekstimuotoiset selitteet, jotka voidaan luoda label-kentän avulla. HTML-esimerkki on esitetty listauksessa 2.2 ja kuvassa 2.5 on esitetty, miltä tämä HTML-esimerkki näyttää selaimessa. [40]



Kuva 2.5. *Form-avainsanan avulla luotu HTML-lomake.*

Toinen vaihtoehto asetuksien syöttämiseen on luvussa 2.1.3 mainittu ESP32-piirin tukema BLE-yhteys, joka on tuettuna myös useimmissa moderneissa puhelimissa. Tämä mahdollistaisi sellaisen mobiilisovelluksen tekemisen, jolla saisi yhteyden infonäyttöön ja saisi vaihdettua sen asetuksia kätevämmin kuin selainta käytettäessä. Tämä vaatisi kuitenkin yhteensopivan puhelimen tai tabletin käyttöä sekä tarkoitukseen sopivan sovelluksen kehittämistä.

3 LAITTEEN TOTEUTUS

Laitteen toteutuksessa hyödynnettiin luvussa 2 käsiteltyä teoriaa. Komponentit valittiin sopivuuden lisäksi myös saatavuuden perusteella. Tämän jälkeen laitteesta rakennettiin prototyyppi, joka ohjelmoitiin vastaamaan luvussa 1 asetettuja tavoitteita. Lisäksi laitteen toimintaa optimoitiin virrankulutuksen kannalta. Optimointia tarkasteltiin erikseen verkkoyhteyksien ja näytön kannalta.

3.1 Kytkentä

Tämän työn elektroniikkapuolen toteutukseen käytettiin EasyEDA-työkalua. Kyseinen työkalu mahdollistaa kytkentöjen ja piirilevyjen suunnittelun selainpohjaisessa ympäristössä. Lisäksi EasyEDA-työkalussa voi simuloida kytkentöjä, mutta sitä ominaisuutta ei käytetty tässä työssä. Työkalu sisältää monia valmiita komponentteja kytkennän ja piirilevyn suunnittelua varten, mutta omien komponenttien lisääminen onnistuu myös käsin. Lisäksi työkalu on kytketty LCSC-verkkokauppaan, joten siitä voi tarkastella suoraan komponenttien datalehtiä ja varastosaldoja. [41]

Piirilevyä suunniteltaessa voi palvelussa käyttää automaattista reititysominaisuutta, joka kytkee piirilevylle sijoitetut komponetit automaattisesti toisiinsa kytkentäkaavion perusteella. Piirilevyn voi myös tilata kätevästi JLCPCB-palvelusta, sillä tämä on myös integroitu EasyEDA-työkaluun. EasyEDA-työkalu on myös ilmainen, mutta piirilevyjen tilaaminen on maksullista. Piirilevysuunnitelman voi kuitenkin halutessaan ladata EasyEDA-työkalusta ja käyttää jotain muuta palvelua sen teettämiseen. [41]

Työtä varten suunniteltiin kytkentä ja sitä vastaava piirilevy. Piirilevy teetettiin käyttäen JLCPCB-palvelua. Piirilevyn päätarkoitus tässä työssä on helpottaa komponenttien valintaa, sillä piirilevylle voi sijoittaa vapaasti haluamiaan komponentteja. Vaihtoehtona tälle oli käyttää valmiita protolevyjä, joiden valikoima on rajallinen. Lisäksi piirilevyn tekeminen mahdollisti kompaktin prototyypin tekemisen, jotta laitteen kehittäminen olisi kätevämpää. Piirilevysuunnittelua ei kuitenkaan käsitellä tässä työssä.

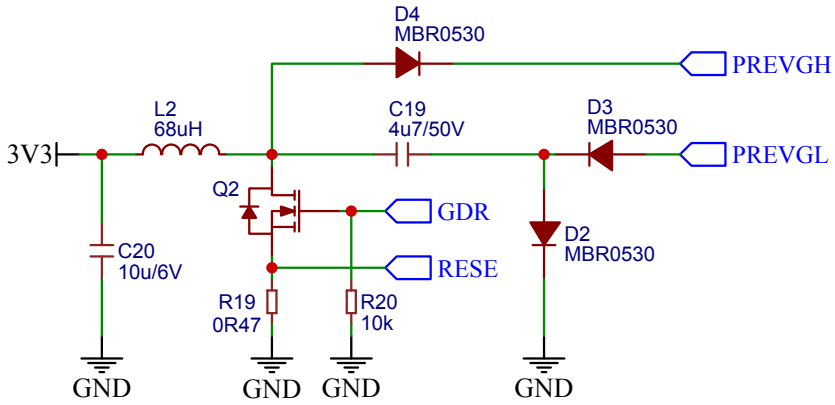
3.1.1 Näyttö

Laitteen näytöksi valittiin lopulta Waveshare Electronicsin 4,2-tuumainen, mustavalkoinen sähköpaperinäyttö [8]. Sähköpaperin valinta perustuu siihen, ettei infonäyttöä ole tarkoitus päivittää usein. Mikäli laitteeseen haluttaisiin esimerkiksi jatkuvasti päivittyvä kello, olisi

luvussa 2.1.1 käsitelty ChLCD-näyttö parempi vaihtoehto. Vastaavan kokoinen ChLCD-näyttö olisi kuitenkin ollut yli kaksi kertaa kalliimpi ja yksi tämän työn tavoitteista oli toteutaa edullinen laite.

Valittu sähköpaperinäyttö on kooltaan noin 9x7 cm ja sen resoluutio on 400x300 pikseliä [8]. Kyseinen resoluutio mahdollistaa sen, että näytölle saadaan mahtumaan riittävä määrä informaatiota, joskin sen lukeminen onnistuu vain melko läheltä näyttöä. Esimerkiksi 8x5 pikselin kirjasintyypillä näytölle saadaan noin 66x37 merkkiä. Tämä on riittävä määrä kalenterin ja useamman tapahtuman esittämiseen, mutta merkkien korkeus on tällöin noin 2 millimetriä. Laitetta varten olisi voinut valita isomman näytön, mutta tämä oli riittävä prototyypin rakentamiseen.

Kyseinen näyttö sisältää ohjainpiirin sekä hakkuripiirin, kuten luvussa 2.1.2 on käsitelty. Näytön käyttöönottoon riittää siis ohjaimen kytkeminen ESP32-piiriin SPI-väylän avulla ja hakkurin vaatimien komponenttien lisääminen kytkentään. Hakkuri on tyypiltään jännitettä nostava hakkuri, sillä 3,3 voltin jännitteestä on tarkoitus muodostaa noin 20 voltin jännite. Lisäyksenä siinä on kytkentä, joka muodostaa myös vastaavan negatiivisen jännitteen. Tämä kytkentä on esitetty kuvassa 3.1. Kuvassa näkyvät PREVGH-, PREVGL-, GDR- ja RESE-signaalit kytketään näyttöön. Näistä PREVGH-signaali on hakkurikytkennän reguloimaton positiivinen jännite ja PREVGL on reguloimaton negatiivinen jännite. GDR-signaali on FET-kytkimen (Field Effect Transistor) ohjaussignaali ja RESE-signaali on tarkoitettu hakkurin virran mittaamiseen.



Kuva 3.1. *Sähköpaperinäytön hakkurikytkentä.*

Kuten luvussa 2.1.5 on käsitelty, vaativat komponentit usein tietyn käyttöjännitteen. Tämän näytön tapauksessa käyttöjännitealue on 2,3 - 3,6 volttia [8], joten näyttö täytyy kytkeä reguloituun jännitteeseen kuten myös ESP32-piiri. Lisäksi kyseinen näyttö vaatii useampia suodatuskondensaattoreita, joita ei ole esitetty tässä erikseen.

3.1.2 Energiavarasto

Työn laite toteutettiin niin, että sen energiavarastona voivat toimia joko kolme sarjaan kytkettyä alkaliparistoa tai yksi litiumioniakku, kuten luvussa 2.1.4 on käsitelty. Käytännössä tämä toimii niin, että piirilevyllä on liitin, johon voidaan kytkeä joko akku tai kolmen pariston

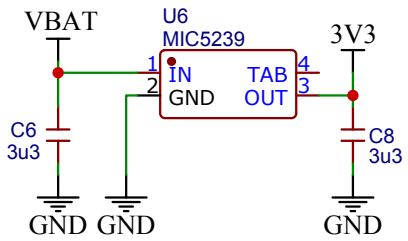
paristokotelo. Paristoiksi valittiin AA-paristot, joiden sähkövaraukseksi oletetaan 2800 milliampeerituntia [12, s. 689]. Tässä työssä laskettavat toiminta-aika-arviot pätevät melko hyvin myös vastaavalla sähkövarauksella varustetulle litiumioniakulle.

AA-paristot valittiin tähän laitteeseen, koska ne ovat yleisesti käytettyjä ja hyvin edullisia. Tällaiseen melko litteään laitteeseen sopisivat paremmin esimerkiksi AAA-paristot, mutta niillä on pienempi noin 1200 milliampeeritunnin sähkövaraus [12, s. 689], joten laitteen toiminta-aika olisi luonnollisesti lyhyempi kuin AA-paristoilla. Vaihtoehtoisesti paristoja voi kytkeä useamman sarjaan, mutta tämä toimii ainoastaan, jos käytössä on hakkuri-regulaattori. Lisäksi hakkuriregulaattorin olisi tuettava isompaa sisäänmenojännitettä. Esimerkiksi neljän sarjaan kytketyn AAA-pariston kokonaisernergiaksi saadaan $(6,0\,\mathrm{V} \cdot 1200\,\mathrm{mAh})/(4,5\,\mathrm{V} \cdot 2800\,\mathrm{mAh}) = 57\,\%$ kolmen AA-pariston kokonaisenergiasta. Neljän pariston kytkeminen sarjaan olisi myös mielekkäämpää kuin kolmen, koska paristoja myydään useimmiten kahden tai neljän kappaleen pakkauksissa.

3.1.3 Regulaattori

Toteutettavan laitteen piirilevy suunniteltiin niin, että siihen on mahdollista laittaa joko lineaariregulaattori tai hakkuriregulaattori. Laitteesta myös rakennettiin kaksi versiota eri regulaattoreilla varustettuna, jotta niiden toimintaa voidaan verrata. Molempien regulaattoreiden vaatimuksena on 3,3 voltin ulostulojännite ja vähintään 500 milliampeerin ulostulovirta [20]. Lisäksi regulaattorien tyhjäkäyntivirran on oltava mahdollisimman pieni.

Lopulta työhön valittiin lineaariregulaattori MIC5239, jonka ulostulojännite on kiinteä 3,3 volttia, ulostulovirta on 500 milliampeeria, kynnysjännite 350 millivolttia ja tyhjäkäyntivirta 23 mikroampeeria [42]. Tämä kynnysjännite tarkoittaa sitä, että paristojen tapauksessa laite lakkaa toimimasta, kun varauksesta on käytetty vain 60%. Tämän voi nähdä kuvasta 2.3. Toimiaakseen vakaasti lineaariregulaattori vaatii vain kaksi kondensaattoria, yksi sisäänmenoon ja yksi ulostuloon. Tämä tekee kytkennästä hyvin yksinkertaisen ja edullisen toteuttaa. Lopullinen kytkentä on esitetty kuvassa 3.2.

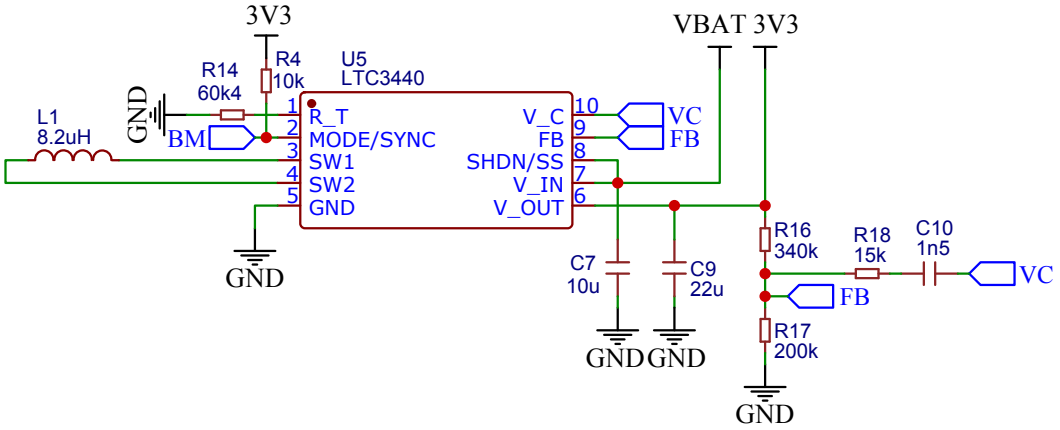


Kuva 3.2. *Lineaariregulaattorin kytkentä.*

Hakkuriregulaattoriksi valittin työssä LTC3440, jonka ulostulovirta on 600 milliampeeria, tyhjäkäyntivirta 25 mikroampeeria ja joka toimii sekä ulostuloa pienemmällä että isommalla sisäänmenojännitteellä. Lisäksi regulaattorin sallittu sisäänmenojännite on 2,5 - 5,5 volttia. Kyseisen hakkuriregulaattorin tyhjäkäyntivirta on pienempi kuin valitun lineaariregulaattorin tyhjäkäyntivirta. Hakkuriregulaattori on kuitenkin asetettava tiettyyn Burst Mode -tilaan, jotta sen tyhjäkäyntivirta olisi näin pieni. Burst mode -tilassa hakkuripiirin ulostulovirta

rajoittuu paljon pienempään arvoon, kuin yllä mainittu 600 milliampeeria. Datalehden mukaan Burst mode -tilassa ulostulovirta saa olla enintään 47 - 60 milliampeeria riippuen sisäänmenojännitteestä. [43]

LTC3440-hakkuripiiri sisältää kaksi FET-transistoria, joilla se voi ohjata siihen kytkettyä kelaa. Tämä tarkoittaa sitä, että hakkuripiirin lisäksi ei tarvita erillisiä FET-transistoreita eikä välttämättä tarvita diodeja. Diodeja voidaan kuitenkin lisätä, jos ulostulojännite ylittää 4,5 voltin rajan tai hakkurikytkennälle halutaan hieman parempi hyötysuhde. Hakkurikytkentä vaatii siis hakkuripiirin lisäksi kelan, sisäänmeno- ja ulostulokondensaattorit, taajuudenvalintavastuksen ja takaisinkytkennän. Taajuudeksi tässä on valittu datalehden suosittelema yhden megahertsin taajuus 60,4 kilo-ohmin vastuksella. Ulostulojännitteen mittaamiseen tarkoitettu takaisinkytkentä muodostuu sen sijaan 340 ja 200 kilo-ohmin vastusjaosta. Lisäksi kytkennässä on vastuksen ja kondensaattorin sarjakytkentä takaisinkytkennän kompensointiin. Lopullinen kytkentä on esitetty kuvassa 3.3. [43]



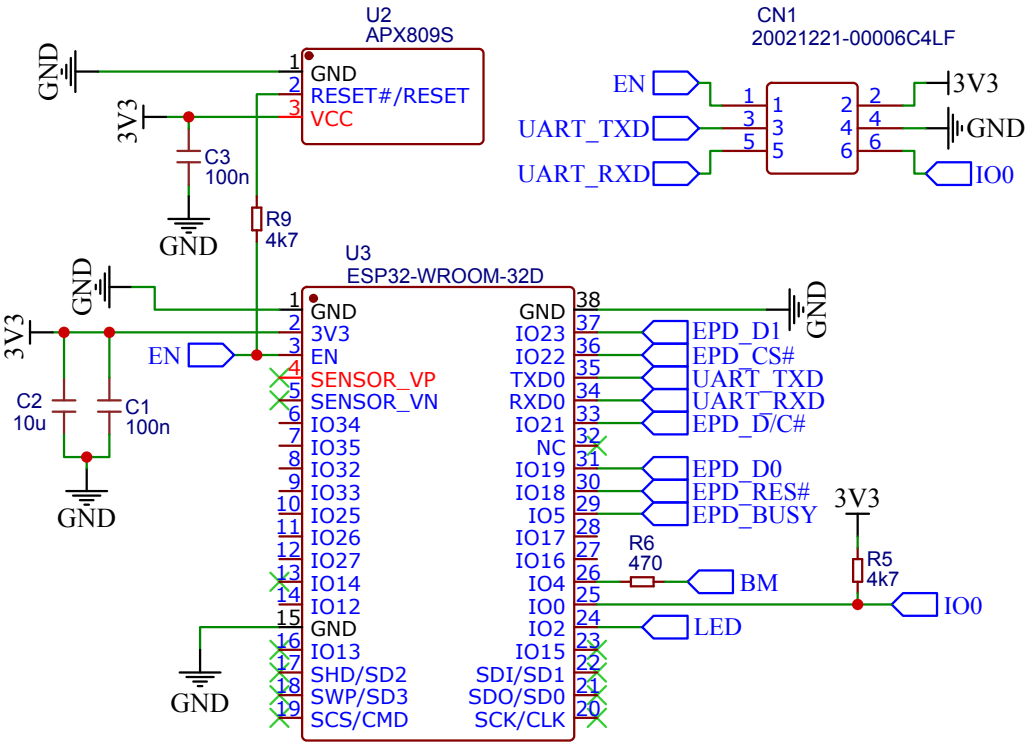
Kuva 3.3. *Hakkuriregulaattorin kytkentä.*

Hakkuripiirin Burst Mode -tilaa on tarkoitus käyttää silloin, kun ESP32-piiri ja näyttö ovat unitilassa. Tätä varten hakkuripiirin MODE/SYNC-sisäänmenoon on kytkettävä käyttöjännite. Vaikkakin ESP32-piirissä on mahdollista ylläpitää ulostulon tasoa Deep-Sleep-unitilan aikana, on helpointa käyttää tässä tapauksessa erillistä ylösvetovastusta [13]. Hakkuripiirin datalehden mukaan tähän käyttötarkoitukseen käy myös hyvin iso vastus, joten kytkentään on valittu 10 kilo-ohmin vastus [43]. Unitilasta herättyään ESP32-piiri ohjaa MODE/SYNC-sisäänmenoa paljon pienemmän vastuksen kautta ja seurauksena on vastusjaon avulla tarpeeksi pieni jännite, jotta hakkuripiiri siirtyy normaaliin toimintatilaan.

3.1.4 ESP32

Laitteen ohjelmoitavaksi osaksi valittiin lopulta ESP32-WROOM-32D-moduuli. Kyseisessä moduulissa on kaksiytiminen ESP32-piiri, flash-piiri, oskillaattori ja sisäänrakennettu antenni. Moduulista valittiin sellainen versio, jossa on 16 megatavun flash-muisti, mutta myös pienempi 4 megatavun versio olisi riittänyt.

ESP32-moduuli itsessään vaatii ainoastaan käyttöjännitteen suodatuskondensaattorit. Tämän lisäksi ESP32-moduuliin on kytketty sähköpaperinäyttö ja ohjelmointiliitin, kuten kuvasta 3.4 voidaan nähdä. Kuvassa on esitetty myös luvussa 3.1.5 käsiteltävä jännitteen valvontapiiri. Kytkennässä näkyy myös LED-signaali (Light Emitting Diode) merkkivaloa varten, sekä BM-signaali luvussa 3.1.3 käsitellyn hakkuripiirin toimintamoodin valintaa varten.



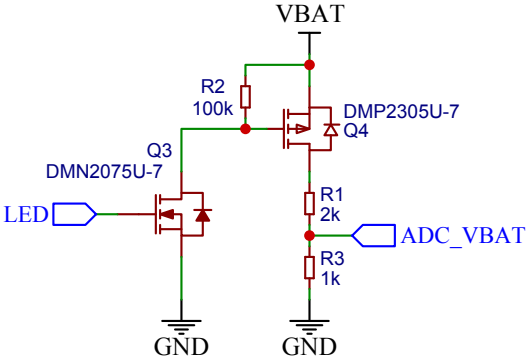
Kuva 3.4. *ESP32-piirin kytkentä.*

Kun laite kytketään päälle, käynnistyy ESP32-piiri tiettyyn tilaan, joka riippuu joidenkin pinnien tilasta [13]. Tässä työssä oleellisia ovat IO0- ja EN-pinnit. Näistä ensimmäinen määrittelee, alkaako piiri suorittamaan ohjelmaa ESP32-WROOM-32-moduulissa olevasta flash-muistista, vai odottaako se ohjelmaa sarjaväylän kautta. EN-pinni sen sijaan toimii ESP32-piirissä reset-pinninä, minkä takia siihen on kytkettynä jännitteen valvontapiiri kuvassa 3.4. Normaalia käynnistystä varten IO0-pinnin ja EN-pinnin täytyy olla kytkettyinä käyttöjännitteeseen. Tätä varten ESP32-piirissä on sisäänrakennettu ylösveto IO0-pinniä varten, mutta tässä työssä on kytketty varmuuden vuoksi ulkoinen ylösvetovastus. Mikäli IO0-pinni on maissa, on ESP32-piiri mahdollista ohjelmoida sarjaväylän kautta.

Kuvasta 3.4 nähdään, että IO0- ja EN-pinnit on kytketty ohjelmointiliittimeen kuten myös sarjaväylän TXD- ja RXD-pinnit (Transmit ja Receive). Tämä mahdollistaa ESP32-piirin automaattisen ohjelmoinnin käyttäen esimerkiksi Espressif Systemsin virallista ohjelmointilaitetta [44]. Toinen vaihtoehto on ohjata IO0- ja EN-pinnejä käsin esimerkiksi painonappien avulla. Sarjaportin kautta on myös mahdollista lukea ESP32-piirin lähettämät vianetsintätulosteet (debug-tulosteet) sopivalla sovelluksella. Tähän tarkoitukseen sopii esimerkiksi tietokonesovellus nimeltä PuTTY [45].

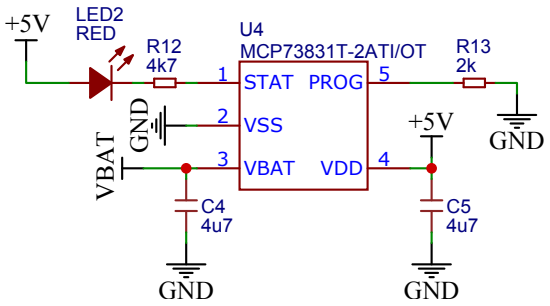
3.1.5 Oheiskytkennät

ESP32-piiri sisältää luvussa 2.1.6 kuvaillun jännitteenvalvontamekanismin [13], mutta siitä huolimatta laitteeseen on päätetty kytkeä erillinen APX809S-jännitteenvalvontapiiri [46]. Piiristä on valittu sellainen versio, joka resetoi siihen kytketyt laitteet, jos käyttöjännite laskee 2,93 voltin alapuolelle. Kyseinen piiri poistaa reset-tilan 240 millisekunnin viiveen jälkeen. Työhön valitun ESP32-WROOM-32D-moduulin datalehden mukaan moduulin käyttöjännitteen alaraja on 3,0 volttia, mutta valittu piiri on todennäköisesti riittävä [20].



Kuva 3.5. *Energiavaraston jännitteenmittauskytkentä.*

Kuten luvussa 2.1.6 on käsitelty, täytyy paristo- tai akkukäytöisessä laitteessa olla energiavaraston jännitteen mittaus. Tämä onnistuu helpoiten mittaamalla paristojen tai akun jännite ESP32-piirin ADC-lohkon avulla. ESP32-piiri sisältää jännitereferenssin ADC-mittauksia varten, mutta sillä ei kuitenkaan voida mitata suoraan piirin käyttöjännitettä isompia jännitteitä. Energiavaraston jännite saadaan laskettua mittausalueelle yksinkertaisella vastusjaolla. Jännitteenmittauskytkennän ei kuitenkaan haluta kuluttavan virtaa jatkuvasti, joten siihen on lisättävä kytkin, jolla mittauskytkennän saa irrotettua energiavarastosta [1]. Työssä käytetty kytkentä on esitetty kuvassa 3.5.



Kuva 3.6. *Litiumioniakun latauspiirin kytkentä.*

Koska laitteesta tehtiin lopulta sellainen, että se voi toimia akun avulla, suunniteltiin kytkentään myös litiumioniakun latauspiiri. Akun lataamiseen valittiin MCP73831-piiri, joka kykenee lataamaan litiumioniakkua maksimissaan 500 milliampeerin virralla. Kyseinen piiri vaatii sisäänmeno- ja ulostulokondensaattorien lisäksi vain yhden vastuksen latausvirran valintaa varten. Tämän lisäksi piirissä on ulostulo, johon voi kytkeä lataustilaa ilmaisevan merkkivalon. Työssä käytetty kytkentä on esitetty kuvassa 3.6, jossa kahden kilo-ohmin vastuksella on valittu 500 milliampeerin latausvirta. [47]

3.1.6 Virrankulutus unitilassa

Laitteen oleellisten komponenttien teoreettinen virrankulutus unitilassa on esitetty taulukossa 3.1. Tämä taulukko on koottu komponenttien datalehdistä ja olettaa ESP32-piirin sekä sähköpaperinäytön olevan unitilassa. Lisäksi hakkuripiirin on oltava Burst Mode -tilassa. Lineaariregulaattorin tapauksessa virrankulutukset voidaan yksinkertaisesti laskea yhteen, koska sitä käytettäessä ei tapahdu energiamuunnosta. Taulukosta 3.1 voidaan siis laskea, että lineaariregulaattoria käytettäessä laitteen virrankulutus unitilassa on $10\,\mu\mathrm{A} + 2\,\mu\mathrm{A} + 10\,\mu\mathrm{A} + 23\,\mu\mathrm{A} = 45\,\mu\mathrm{A}$.

Taulukko 3.1. *Laitteen komponenttien virrankulutukset unitilassa.*

Komponentti	Virrankulutus (uA)
ESP32	10
EPD	2
Jännitteen valvontapiiri	10
Lineaariregulaattori	23
Hakkuriregulaattori	25

Hakkuriregulaattorin tapauksessa sen sijaan tapahtuu energiamuunnos, joten se on otettava huomioon. Kytkennästä tiedetään, että regulaattorin ulostulo on 3,3 volttia ja taulukosta 3.1 nähdään, että komponenttien yhteenlaskettu virrankulutus on 22 mikroampeeria, kun itse regulaattoria ei oteta huomioon. Hakkuriregulaattorin datalehdeltä taas nähdään, että hakkuripiirin hyötysuhde Burst Mode -tilassa on noin 75%, minkä avulla voidaan laskea koko kytkennän virrankulutus [43]. Kytkennän tarvitsema teho saadaan kaavasta $P_{out} = UI = 3,3\,\mathrm{V} \cdot 22\,\mu\mathrm{A} = 75\,\mu\mathrm{W}$. Seuraavaksi on huomioitava regulaattorin hyötysuhde kaavalla $P_{in} = P_{out}/\eta = 75\,\mu\mathrm{W}/0,75 = 100\,\mu\mathrm{W}$. Tämän jälkeen voidaan laskea sisäänmenovirta olettaen, että sisäänmenojännite on 4,5 volttia ja huomioiden hakkuriregulaattorin tyhjäkäyntivirta. Tämä onnistuu kaavalla $I_{in} = P_{in}/V_{in} + I_q = 100\,\mu\mathrm{W}/4,5\,\mathrm{V} + 25\,\mu\mathrm{A} = 46,5\,\mu\mathrm{A}$.

3.2 Ohjelmisto

Tässä työssä toteutettavan laitteen ohjelmointi suoritettiin käyttäen PlatformIO-pakettia ja Visual Studio Code -editoria. PlatformIO sisältää ESP32-piirin ohjelmointiin tarvittavat ESP-IDF-kirjastot ja työkalut. Visual Studio Code tarjoaa alustan PlatformIO-paketin käyttämiseen ja ohjelman kirjoittamiseen. Nämä ovat myös ilmaisia työkaluja. [22][48][49]

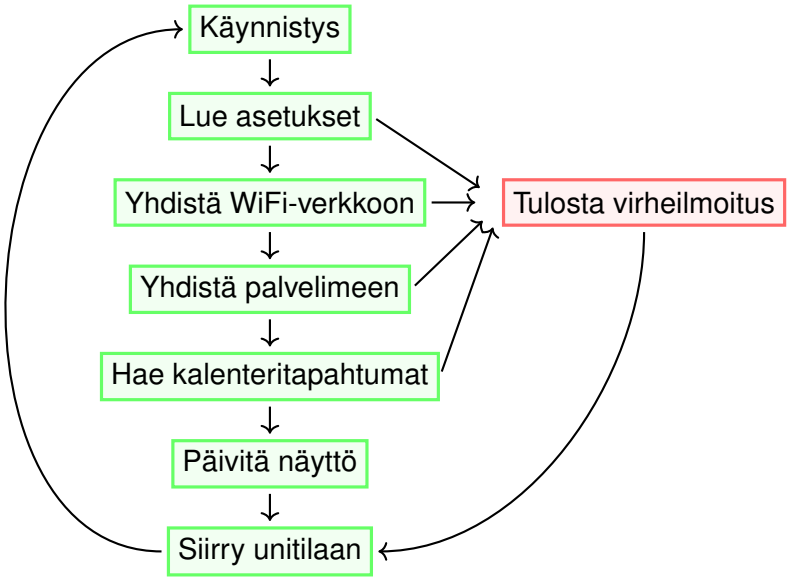
ESP32-piirin fyysiseen ohjelmointiin käytettiin Espressif Systemsin virallista ohjelmointilaitetta nimeltä ESP-Prog. Kyseinen laite mahdollistaa ESP32-piirin ohjelmoinnin sekä ESP32-piirin debug-tulosteiden lukemisen, sillä ohjelmointilaite näkyy tietokoneelle tavallisena sarjaporttina. Lisäksi laitteessa on JTAG-liityntä (Joint Test Action Group), jolla ESP32-piirin toimintaa voi seurata tarkemmin. JTAG-mahdollisuutta ei kuitenkaan käytetty tätä työtä tehdessä. [44]

ESP-IDF sisältää kirjaston debug-tulosteiden tekemiseen. Kyseinen kirjasto sisältää funktiot muun muuassa virheiden, varoitusten ja informatiivisten viestien tulostamiseen. Tämän lisäksi käännösvaiheessa voidaan rajoittaa, mitkä näistä tulosteista sisällytetään käännettyyn ohjelmaan. Tällä tavoin voidaan määritellä, että näkyviin tulevat vain sellaiset tulosteet, joita kehittäjä sillä hetkellä tarvitsee. Lisäksi näissä debug-tulosteissa on värikoodaus, jonka avulla esimerkiksi virheilmoitukset tulostuvat punaisella värillä. [15]

Sähköpaperinäytön ohjelmointiin käytettiin apuna näytön valmistajan tarjoamia ohjelmointiesimerkkejä, jotka ovat saatavilla GitHub-palvelussa. Tarkemmin ottaen ohjelman kirjoittamiseen käytettiin palvelusta löytyviä Arduino-esimerkkejä, koska ESP32-piirille ei ollut valmiita esimerkkejä. [50]

3.2.1 Ylemmän tason toiminta

Laitteen näytöksi valittiin sähköpaperi, minkä seurauksena laite voi olla suurimman osan ajasta virransäästötilassa. Luvussa 2.2.5 esitetyistä ESP32-piirin unitiloista tähän käyttötarkoitukseen sopii parhaiten Deep-Sleep-unitila, koska tässä tapauksessa riittää, että laite osaa herätä itse unitilasta ja että siinä on jokin muisti tietojen tallentamiseen. Tietojen tallentaminen ei olisi myöskään välttämätöntä, mutta sen tarpeellisuutta on avattu tarkemmin luvussa 3.3.2.



Kuva 3.7. *Tilakaavio ohjelman ylemmän tason toiminnasta.*

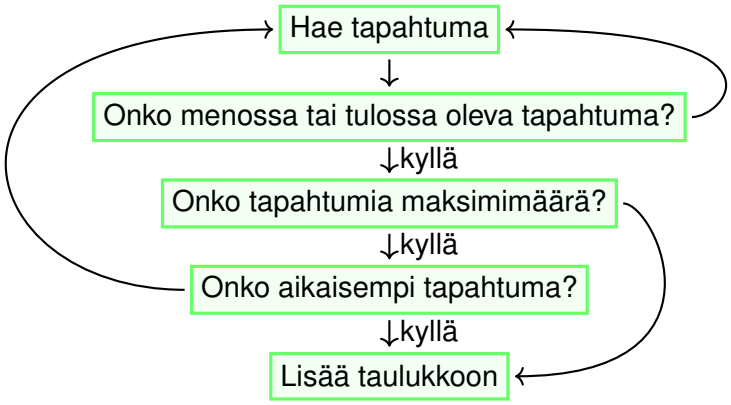
Valitun unitilan takia laite herää aina kuin ensimmäistä kertaa, sillä sen käyttömuisti tyhjenee unitilassa. Tämä tarkoittaa sitä, että laitteen toiminnasta voidaan tehdä hyvin suoraviivainen kuten kuvassa 3.7 on esitetty. Käynnistymisen tai unitilasta heräämisen jälkeen laite yrittää ensin lukea tarvittavat asetukset flash-muistista. Asetuksien tarkempi luonne on selitetty luvussa 3.2.3. Mikäli asetuksia ei ole tai ne ovat korruptoituneet, tulostaa laite näytölleen virheilmoituksen ja siirtyy takaisin unitilaan.

Jos laitteen asetukset ovat muistissa, yhdistää laite seuraavaksi WiFi-verkkoon. Mikäli tämä epäonnistuu esimerkiksi väärien asetuksien takia, tulostaa laite virheilmoituksen ja siirtyy takaisin unitilaan. Onnistuneen WiFi-verkkoon yhdistämisen jälkeen laite yrittää yhdistää sähköisen kalenterin palvelimeen. Jos tämä epäonnistuu, tulostaa laite taas virheilmoituksen ja siirtyy unitilaan. Mikäli yhteyden muodostaminen palvelimeen onnistuu, hakee laite kalenteritiedot, päivittää näytön ja siirtyy unitilaan. Kalenteritietojen hakeminen saattaa myös epäonnistua vaikka yhteyden muodostaminen palvelimeen onnistuisikin. Tällöin on myös hyvä tulostaa virheilmoitus ja siirtyä unitilaan. Muussa tapauksessa on mahdollista, että vain osa kalenteritiedoista saadaan haettua ja kaikki tarvittava informaatio ei tule näytölle.

WiFi-yhteyden muodostaminen tapahtuu käytännössä niin, että ensin ESP32-piirin WiFi-radio kytketään päälle sopivia funktiokutsuja käyttäen. Tämän jälkeen asetetaan laite toimimaan WiFi-päätelaitteena. Seuraavaksi asetetaan verkon nimi ja salasana ja käynnistetään verkkoonyhdistämisprosessi. Tässä vaiheessa laite skannaa eri WiFi-kanavia kunnes se löytää, mitä kanavaa haluttu tukiasema käyttää, kuten luvussa 2.2.1 on selitetty. Tämän jälkeen laite yrittää yhdistää kyseiseen WiFi-verkkoon ja pyytää tukiasemalta IP-osoitteen. IP-osoitteen saatuaan laite on valmis hakemaan kalenteritietoja.

3.2.2 Kalenteritietojen haku

IP-osoitteen saatuaan laite on valmis yhdistämään palvelimeen, josta kalenteritiedot on tarkoitus hakea. Tämä onnistuu jälleen sopivia funktiokutsuja käyttäen. Koska laitteen on tarkoitus hakea kalenteritietoja useammasta eri lähteestä, yhdistää laite vuorotellen laitteeseen tallennettuihin palvelimiin. Palvelinyhteyden muodostamisen jälkeen voi laite lähettää HTTP-kutsun, johon palvelin vastaa, kuten luvussa 2.2.3 on esitetty. Mikäli kaikki menee oikein, vastauksessa on mukana halutut kalenteritiedot.



Kuva 3.8. *Kalenteritietojen haun tilakaavio.*

Kaikkien palvelimelta haettavien kalenteritietojen tallentaminen laitteen muistiin ei ole mahdollista, sillä laitteen muisti on rajallinen. Lisäksi laite voi esittää näytöllään vain rajallisen määrän tapahtumia, vaikka kaikki tapahtumat saisikin tallennettua muistiin. Tämän

takia on rajattava, mitkä kalenteritapahtumat tallennetaan ja mitkä jätetään huomiotta. Kalenterin tapauksessa on mielekästä esittää esimerkiksi kuluvan viikon tapahtumat, tai pelkästään tulevat ja menossa olevat tapahtumat. Lisäksi kalenteritiedoista poimitaan oleelliset tiedot ja esimerkiksi tapahtumien muokkausaikaleimat voidaan jättää huomiotta.

Kuvassa 3.8 on esitetty eräs toteutus, jolla voidaan hakea tietty määrä menossa tai tulossa olevia tapahtumia. Tässä toteutuksessa kalenteritietoja käsillään yksi tapahtuma kerrallaan sitä mukaa, kun palvelin niitä lähettää. Yhden kalenteritapahtuman löydettyään algoritmi tarkastelee, onko se tuleva tai menossa oleva tapahtuma ja jättää sen huomiotta, mikäli se ei ole. Tämän jälkeen tarkastellaan, onko kalenteritapahtumia jo tallennettuna maksimimäärä. Jos kalenteritapahtumia ei ole maksimimäärä, tallennetaan tapahtuma muistiin. Mikäli kalenteritapahtumia on jo maksimimäärä, etsitään muistista myöhäisin tapahtuma ja tarkistetaan, onko se myöhempi kuin lisättävä tapahtuma. Jos on, korvataan myöhempi tapahtuma juuri haetulla, muutoin haettu tapahtuma jätetään huomiotta. Tämän jälkeen algoritmi hakee seuraavaa tapahtumaa.

3.2.3 Asetukset

Laitteen asetuksien syöttäminen on toteutettu HTTP-palvelimen avulla, kuten luvussa 2.2.6 on käsitelty. Tätä varten laitteessa on erillinen painonappi, jonka painamisen jälkeen laite herää unitilasta ja siirtyy toimimaan WiFi-tukiasemana. Tämän jälkeen laite toimii HTTP-palvelimena, jonka avulla laite voidaan konfiguroida. Koska kyse on paristokäyttöisestä laitteesta, täytyy sen siirtyä unitilaan tietyn ajan kuluttua, mikäli käyttäjä ei tee mitään. Muussa tapauksessa laitteen paristot tai akku tyhjenevät hyvin nopeasti.

Kun käyttäjä on syöttänyt HTML-sivulle laitteen asetukset ja hyväksynyt ne, täytyy nämä asetukset tallentaa laitteen flash-muistiin. Tämä onnistuu käyttäen sopivia ESP-IDF-kirjaston funktioita. Ensiksi flash-kirjasto on alustettava. Tämän jälkeen kukin asetus kirjataan avain-tieto-pareja käyttäen. Lopuksi kirjatut tiedot kirjoitetaan flash-muistiin. Flash-muistin tekniikasta johtuen tietoja ei kirjoiteta yksitellen suoraan muistiin.

WiFi-asetusten tallentamiseen on olemassa valmiita funktioita, joten niitä ei toteutettu erikseen tässä työssä. Lisäksi valmiit funktiot tallentavat muutakin informaatiota kuin tukiaseman nimen ja salasanan kuten luvussa 3.3.1 on käsitelty. Valmiista funktioista huolimatta, WiFi-asetuksien olemassaolo flash-muistissa täytyy tarkistaa kuten muidenkin asetusten tapauksessa.

3.2.4 Unitila

Kuten luvussa 3.2.1 on käsitelty, on laitteessa päätetty käyttää Deep-Sleep-unitilaa. Tähän unitilaan siirtyminen onnistuu sopivan funktiokutsun avulla. Ennen tätä on kuitenkin määriteltävä, miten laite herää Deep-Sleep-unitilasta. Laitteen on tarkoitus toimia pääasiassa autonomisesti, joten herääminen on toteutettava ajastimella. Tässä tapauksessa on päätetty käyttää 15 minuutin ajastinta, koska tiheämpi päivitystahti tarkoittaisi suurempaa

virrankulutusta. Harvempi päivitystahti sen sijaan tarkoittaisi sitä, että jokin lyhyellä varoitusajalla lisätty tapahtuma voisi jäädä käyttäjältä huomaamatta.

Laite on kuitenkin tarkoitus pystyä päivittämään myös käsin. Tätä varten laitteessa on painonappi, jota painamalla laite herää unitilasta samalla tavalla kuin ajastimen tapauksessa, hakee uudet kalenteritiedot ja päivittää näytön. Kyseinen painonappi on myös asetettava menetelmäksi unitilasta heräämiseen ennen unitilaan siirtymistä. Laitteessa on oltava myös toinen nappi, jotta laite voidaan herättää asetuksien syöttämistä varten, kuten luvussa 3.2.3 on esitetty.

Luvussa 2.2.5 on myös käsitelty, että laite menettää käyttömuistinsa sisällön, kun se siirtyy Deep-Sleep-unitilaan. Tämä ei kuitenkaan ole laitteen kannalta ongelma, koska laitteella ei ole tarvetta säilyttää päivitysten välissä muuta informaatiota, kuin flash-muistissa olevat asetukset. Haettujen kalenteritapahtumien säilyttäminen unitilan yli olisi järkevää siinä tapauksessa, jos laite ei saa haettua uusia kalenteritapahtumia jonkin virheen seurauksena. Tällöin voidaan kuitenkin käyttää näytön osittaispäivittämistä, eli viime kerralla näytölle kirjoitettuja tapahtumia ei pyyhitä pois. Tässä tapauksessa esimerkiksi näytön alareunaan voi tulla ilmoitus, joka kertoo, että kalenteritietojen haku epäonnistui.

Laitteen ollessa Deep-Sleep-unitilassa on kuitenkin eräs muistilohko, jonka sisältö ei tyhjene. Kyseessä on RTC-lohkon muisti, joka olisi tarpeeksi iso kalenteritapahtumien tallentamiseen. Tästä huolimatta edellä mainittu näytön osittain päivittäminen on parempi vaihtoehto virrankulutuksen kannalta. RTC-lohkon muistia voisi kuitenkin käyttää ajan arviointiin. Laite ei nimittäin tiedä kellonaikaa, jos se ei esimerkiksi saa yhteyttä WiFi-tukiasemaan. Tällöin se voisi arvioida kellonaikaa viimeisen päivityksen kellonajan ja unitilassa vietetyn ajan perusteella. Tämä onnistuu, mikäli laite herää unitilasta ajastimen eikä napinpainalluksen avulla. Tätä arvioitua kellonaikaa voisi käyttää esimerkiksi virheilmoituksen tulostamiseen. Tulosteesta näkisi, että laite edelleen toimii vaikkei se saisikaan haettua kalenteritietoja.

3.3 Toiminnan optimointi

Luvuissa 3.1 ja 3.2 toteutetun laitteen toimintaa on syytä tarkastella lähemmin, koska laitteen on tarkoitus toimia paristojen tai akun varassa mahdollisimman pitkään. Verkkoon kytketyn laitteen tapauksessa on monia tekijöitä, joihin tässä ei voida vaikuttaa. Tällaisia ovat esimerkiksi haettavien kalenterien määrä, niiden koko, verkon nopeus sekä esimerkiksi WiFi-verkon signaalin taso. On kuitenkin olemassa asioita, joihin tässä työssä voidaan vaikuttaa.

3.3.1 Verkkoyhteyksien optimointi

Listauksessa 3.1 on esitetty osa laitteen debug-tulosteista normaalin kalenteritietojen haun ajalta. Nämä tulosteet koostuvat tyypistä, aikaleimasta, tunnisteesta ja itse viestistä. Tässä tapauksessa tyyppi on iso i-kirjain, joka tarkoittaa informaatioviestiä. Muita vaihtoehtoja

ovat esimerkiksi iso e-kirjain, joka tarkoittaa virhettä sekä iso w-kirjain, joka tarkoittaa varoitusta. Seuraavana tulosteissa on aikaleima, joka on esitetty millisekunneissa. Tämän jälkeen tulosteessa on tunniste, joka ilmaisee debug-tulosteen lähdettä. Tässä tapauksessa wifi- ja event-tunnisteet viittaavat ESP-IDF-kirjaston sisäisiin funktioihin ja main-tunniste on käyttäjän määrittelemä. Lopuksi tulosteessa on käyttäjän tai jonkin kirjaston määrittelemä viesti. [15]

```
I (408) main: starting WiFi
I (2638) wifi: connected with xxx, channel 11, HT20, bssid = aa:2e:4d:8a:8c:ee
I (3228) event: sta ip: 192.168.43.230, mask: 255.255.255.0, gw: 192.168.43.215
I (3228) main: fetching calendar data
I (5558) main: connection established...
I (5558) main: 198 bytes written
I (6318) main: 21502 bytes read
I (6318) main: connection closed
I (9028) main: connection established...
I (9028) main: 194 bytes written
I (10368) main: 68720 bytes read
I (10368) main: connection closed
I (12888) main: connection established...
I (12888) main: 222 bytes written
I (13398) main: 7701 bytes read
I (13398) main: connection closed
I (18758) main: entering deep sleep
```

Listaus 3.1. *ESP32-ohjelman debug tulosteet ilman optimointia.*

Listauksesta 3.1 nähdään, että ensimmäisen ja toisen tulosteen välillä kuluu aikaa vähän yli kaksi sekuntia. Kuten luvussa 2.2.1 on käsitelty, ESP32-piiri skannaa tässä vaiheessa WiFi-verkkoja ja etsii haluttua tukiasemaa. Yksi tapa nopeuttaa laitteen toimintaa on tallentaa WiFi-asetusten mukana myös kanavan numero. Seuraavalla yhdistämiskerralla tukiasema käyttää todennäköisesti edelleen samaa kanavaa, joten listauksessa näkyvä skannaaminen on tarpeetonta. Toisen ja kolmannen tulosteen välillä kuluvaan noin 700 millisekunnin aikaan ei sen sijaan voida vaikuttaa, sillä se kuluu siihen, että tukiasema antaa laitteelle IP-osoitteen.

Seuraavaksi listauksesta 3.1 nähdään, että palvelimeen yhdistämiseen kuluu yli sekunti aikaa. Lisäksi palvelimeen yhdistetään aina uudestaan jokaista kalenteria varten, kuten luvussa 3.2.2 on käsitelty. Koska tässä työssä on päätetty käyttää ainoastaan Google Calendar -palvelua, voidaan yhteys jättää auki, kunnes kaikkien kalenterien kalenteritiedot on haettu. Tämä onnistuu esimerkiksi käyttämällä HTTP/1.1-yhteyttä HTTP/1.0-yhteyden sijaan [51, s. 44]. HTTP/1.1-yhteys jää oletuksena auki siihen asti, kunnes se erikseen suljetaan. Käytännössä tämä onnistuu niin, että HTTP/1.0 teksti korvataan tekstillä HTTP/1.1 GET-kyselyssä. Tämän lisäksi on erikseen tarkasteltava, koska pyydetty tieto on saapunut ja osattava lähettää seuraava HTTP-kysely sekä lopussa sulkea yhteys.

Tämän lisäksi listauksesta 3.1 nähdään, että näytön päivittämiseen kuluu huomattava aika kaikkeen muuhun toimintaan nähden. Tulosteesta nähdään myös, että WiFi-yhteyttä ei suljeta koskaan vaikka palvelinyhteys suljetaankin. WiFi-yhteyden auki pitäminen vaatii paljon enemmän energiaa kuin näytön päivittäminen, joten se kannattaa sulkea, kun sitä ei enää tarvita. Myöskään ESP32-piirin ei tarvitse olla silloin aktiivisena ja se voi olla esimerkiksi luvussa 2.2.5 mainitussa Light-Sleep-unitilassa. Light-Sleep-unitila on sellainen, josta prosessori voi herätä nopeasti ja jatkaa ohjelman suoritusta siitä mihin se jäi. Tämän takia se soveltuu hyvin tilanteisiin, joissa lepotilan kesto on melko lyhyt.

```
I (413) main: starting WiFi
I (1423) wifi: connected with xxx, channel 2, HT20, bssid = 2c:4d:54:1c:3e:94
I (2233) event: sta ip: 192.168.1.162, mask: 255.255.255.0, gw: 192.168.1.1
I (2233) main: fetching calendar data
I (4253) main: connection established...
I (4253) main: 198 bytes written
I (4913) main: 21538 bytes read
I (4913) main: 194 bytes written
I (5683) main: 68780 bytes read
I (5683) main: 222 bytes written
I (5993) main: 7737 bytes read
I (5993) main: connection closed
I (6033) main: stopping WiFi
I (11383) main: entering deep sleep
```

Listaus 3.2. *ESP32-ohjelman debug-tulosteet verkko-optimointien kanssa.*

Listauksessa 3.2 on esitetty laitteen toiminta edellä mainittujen optimointien kanssa. Listauksesta nähdään, että WiFi-yhteyden muodostaminen kestää nyt vain sekunnin verran kahden sekunnin sijaan. Lisäksi palvelinyhteys muodostetaan vain kerran ja kaikkien kalenteritietojen hakemiseen kuluu vähän alle neljä sekuntia, kun taas listauksessa 3.1 kaikkien kalenteritietojen hakemiseen kului kymmenen sekuntia. Lisäksi listauksesta 3.2 nähdään, että WiFi-yhteys on suljettu kalenteritietojen hakemisen jälkeen.

3.3.2 Näytön optimointi

Listauksessa 3.3 on esitetty näytön päivittämiseen liittyvät debug-tulosteet. Listauksesta voidaan nähdä kolme eri vaihetta. Ensimmäisenä on näytönohjaimen alustus, joka kestää noin puoli sekuntia. Tämä sisältää muun muuassa piirin resetoinnin ja hakkurikytkennän käynnistämisen. Tämän jälkeen kuvatieto siirretään näytönohjaimen muistiin, missä kestää myös noin puoli sekuntia. Seuraavaksi ohjain piirtää näytölle muistiin siirretyn kuvan, missä kestää noin neljä sekuntia.

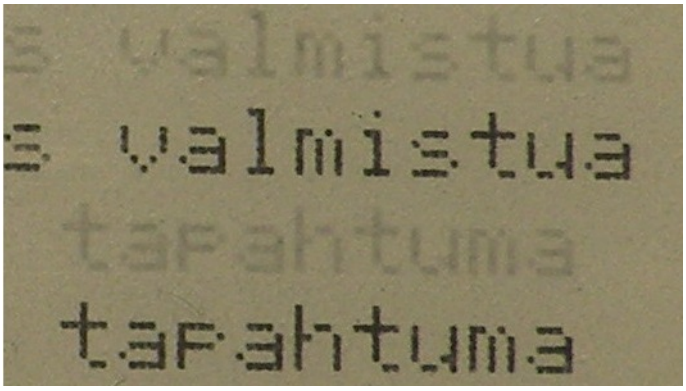
Ensimmäinen ongelma on sähköpaperinäytön resetointiin tarkoitettu, 400 millisekuntia kestävä sekvenssi. Tätä ei ole määritelty näytön datalehdessä, mutta valmistajan käyttöesimerkissä on tehty näin. Kyseinen resetointi voidaan kuitenkin tehdä sillä aikaa, kun laite odottaa esimerkiksi WiFi-yhteyden muodostumista. Tämä perustuu oletukseen, että

näyttöä ei käytetä 400 millisekunnin sisällä laitteen käynnistyksestä. Se onnistuu helpoiten niin, että laitteen käynnistyessä näytön reset-signaalia ohjaava pinni asetetaan aktiiviseen tilaan. Tämän jälkeen asetetaan 200 millisekunnin ajastin, jonka jälkeen reset-signaalia ohjaava pinni otetaan pois aktiivisesta tilasta. Ajastimen aikana laite voi suorittaa muuta ohjelmaa. Ajastimen jälkeen kuuluisi odottaa vielä 200 millisekuntia, mutta sitä ei tarvitse erityisesti odottaa, kun tiedetään, että laite on vielä hakemassa kalenteritietoja.

```
I (6063) graphics: Starting EPD
I (6563) graphics: EPD inited
I (6563) epd4in2: Uploading picture data
I (6983) epd4in2: Upload complete
I (11183) graphics: EPD updated
I (11383) graphics: EPD sleeping
```

Listaus 3.3. *Näytön päivittämiseen liittyvät debug-tulosteet ilman optimointia.*

Toinen ongelma on kuvatiedon siirtäminen näytönohjaimen muistiin. Tätä vaihetta ei voida suorittaa etukäteen, koska ennen näytön päivittämistä on saatava haettua kalenteritiedot. Tätä vaihetta voidaan kuitenkin nopeuttaa käyttämällä luvussa 2.2.4 mainittua DMA-lohkoa. Tämä onnistuu niin, että yksittäisten tavujen sijaan SPI-lohkon ohjaamisesta vastaavalle funktiolle välitetään esimerkiksi tuhat tavua kerrallaan. Tämä vähentää kuvatiedon siirtämistä varten suoritettavan ohjelman määrää ja täten nopeuttaa tiedonsiirtoa.



Kuva 3.9. *Sähköpaperinäytön nopean päivityksen seurauksena näkyvä haamukuva.*

Näyttöä voidaan kuitenkin nopeuttaa entisestään. Datalehdellä ilmoitettu noin kahdeksan sekunnin päivitysaika perustuu siihen, että näytön täydellinen päivittäminen vaatii monimutkaisen jännitekäyrän, jotta näyttö ei vaurioidu [10]. Täydellisen päivittämisen sijaan voidaan kuitenkin tehdä niin sanottu nopea päivitys, jota ei kuitenkaan ole annettu valmistajan esimerkeissä tässä työssä käytettävälle näytölle. Tällainen nopea päivitys löytyy esimerkiksi saman valmistajan 2,9-tuuman näytön esimerkistä. Eräs Ben Krasnow on kuitenkin kehittänyt tällaisen nopeamman päivitysprosessin tässä työssä käytettävälle näytölle käyttäen hyväksi muun muuassa toisen näyttövalmistajan datalehteä [11]. Tämän muunnetun prosessin avulla näytön päivittäminen onnistuu huomattavasti nopeammin. Haittapuolena tässä on se, että näytön päivittäminen ei onnistu täydellisesti ja kuvaan jää haamukuvia, joista esimerkki on näytetty kuvassa 3.9.

Edellä käsiteltyjen näytön optimointien tulokset voi nähdä listauksesta 3.4. Listauksesta nähdään, että nyt näytönohjaimen käynnistämiseen kuluu vain 100 millisekuntia, kuvan siirtämiseen vain 10 millisekuntia ja kuvan päivittämiseen vain 400 millisekuntia edellisen neljän sekunnin sijaan. Näytönohjaimen siirtyminen unitilaan kestää edelleen 200 millisekuntia ja sille ei voida tehdä mitään. ESP32-piiri voi kuitenkin olla Light-Sleep-unitilassa näytön päivittämisen aikana, kuten luvussa 3.3.1 on käsitelty. ESP32-piirin on oltava aktiivinen ainoastaan kuvan siirtämiseen kuluvan 10 millisekunnin ajan.

```
I (7693) graphics: Starting EPD
I (7793) graphics: EPD inited
I (7793) epd4in2: Uploading picture data
I (7803) epd4in2: Upload complete
I (8203) graphics: EPD updated
I (8403) graphics: EPD sleeping
```

Listaus 3.4. *Näytön päivittämiseen liittyvät debug-tulosteet optimointien kanssa.*

Kuten luvussa 2.1.2 on käsitelty, on näyttö mahdollista päivittää myös osittain. Tämä on järkevää silloin, kun verkosta haetut kalenteritiedot eivät ole muuttuneet edellisen päivityskerran jälkeen. Siinäkin tapauksessa kannattaa kuitenkin päivittää osa näytöstä niin, että käyttäjä näkee, että laite on hakenut uusimmat kalenteritiedot, eikä ole jäänyt jumiin. Tässä tapauksessa riittää pieni tilapalkki, jossa kerrotaan, koska kalenteritiedot on viimeksi haettu. Kyseinen tilapalkki voi myös esittää mahdolliset virheilmoitukset. Tällaista näytön osittain päivittämistä ei ole myöskään esitetty näytön valmistajan esimerkissä, mutta näytön datalehdeltä selviää, että sellainen mahdollisuus on olemassa [8].

Näytön päivittäminen osittain edellyttää sitä, että laite tietää, etteivät kalenteritiedot ole muuttuneet edellisen päivityskerran jälkeen. Tämän selvittäminen onnistuisi helpoiten, jos palvelimelta saataisiin suoraan kyseinen tieto jokaista kalenteria kohden. Tämä poistaisi myös tarpeen hakea kalenteritiedot kokonaan ja pienentäisi virrankulutusta entisestään. Vaihtoehtona on esimerkiksi HTTP-protokollan Content-MD5-, ETag- tai Last-Modified-tieto. Näistä Content-MD5 on tarkistussumma, ETag on eräänlainen versiotieto ja Last-Modified ilmaisee, koska tietoja on viimeksi muokattu. [51]

Valitettavasti Google Calendar -palvelin ei kuitenkaan lähetä mitään edellä mainituista, eli ainoa vaihtoehto on ladata kalenteritiedot kokonaan. Tässäkin tapauksessa on kuitenkin mahdollista laskea tarkistussumma itse ja tallentaa se laitteen muistiin. Tällä tavalla kalenteritietojen hakemisen jälkeen laite voi verrata niiden tarkistussummaa muistissa olevaan edellisen päivityskerran tarkistussummaan ja olla päivittämättä näyttöä, mikäli nämä täsmäävät. Tämä tarkistussumma on järkevä tallentaa luvussa 2.2.5 mainittuun RTC-lohkon muistiin, mutta sen tallentaminen flash-muistiin on myös mahdollista.

Näytön nopeampi päivittäminen ja osittain päivittäminen voivat kuitenkin vaurioittaa näyttöä, mikäli näyttöä päivitetään vain näillä menetelmillä [10]. Näyttö on luonnollisesti päivitettävä kokonaan silloin, kun kalenteritiedot muuttuvat tai jokin muu näytöllä oleva tieto muuttuu. Tämän lisäksi laitteeseen on tehtävä laskuri, joka varmistaa, että näyttö päivitetään

kokonaan tietyin väliajoin, esimerkiksi viiden nopean päivityksen välein. Kyseinen laskuri voisi sijaita myös RTC-lohkon muistissa.

3.3.3 Muut optimoinnit

Laitteen toiminta-aikaa on mahdollista optimoida vielä enemmän, mutta tämä tapahtuu käytettävyyden kustannuksella. Eräs vaihtoehto on rajata, mihin kellonaikaan näyttö päivittyy. Jos laitetta käytetään esimerkiksi työpaikalla, on sen toiminta järkevää rajoittaa työaikaan. Tätä varten laitteen asetuksissa voi olla kohta, jossa voi asettaa päivitystiheyden kullekin aikavälille. Tällä tavalla laitteen saa päivittymään harvemmin silloin, kun tiheämmälle päivitystahdille ei ole tarvetta. Esimerkiksi näytön päivittäminen ainoastaan kello 8 - 16 kolminkertaistaisi toiminta-ajan. Näytöstä olisi silti luettavissa, milloin se on viimeksi hakenut uusimmat kalenteritiedot luvussa 3.3.2 kuvatun tilapalkin avulla. Toiminta-aikarajoituksen ei kuitenkaan tarvitse olla kiinteä, vaan se voi olla käyttäjän valittavissa laitteen asetussivulla.

Toinen mahdollinen kompromissi on olla päivittämättä näyttöä lainkaan, jos kalenteritiedoissa ei ole tapahtunut mitään muutosta. Tällä tavalla laite olisi aktiivisessa tilassa noin sekunnin verran vähemmän verrattuna näytön nopeaan päivitykseen. Tällöin käyttäjä ei kuitenkaan saisi tietää, onko laite hakenut uusimmat kalenteritiedot vai ei, ellei käyttäjä satu katsomaan laitetta juuri sillä hetkellä, kun laite on hakemassa kalenteritietoja ja laitteessa palaa merkkivalo. Laite voisi esimerkiksi olla jumissa, eikä käyttäjä saisi tietää tästä mitenkään. Lisäksi näytön päivittämiseen kuluva energia on huomattavasti pienempi kuin kalenteritietojen hakemiseen kuluva energia, joten tällä on hyvin vähäinen merkitys optimoinnin kannalta.

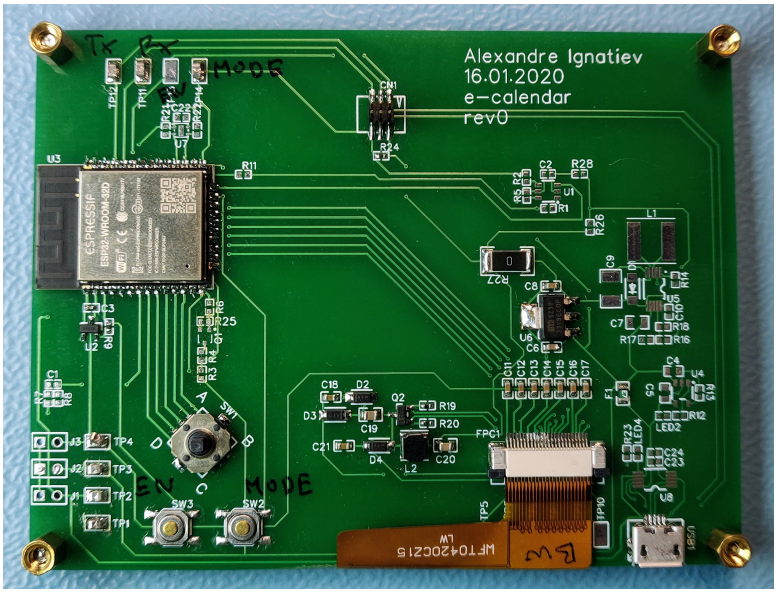
Kuten luvussa 3.3.2 on mainittu, HTTP-protokollan Content-MD5-, ETag- ja Last-Modified-tietoja ei voida käyttää sen selvittämiseen, ovatko kalenteritiedot muuttuneet palvelimella. Näiden sijaan on mahdollista käyttää kalenteritietojen kokoa. Kalenteritietojen muuttuessa myös niiden koko todennäköisesti muuttuu. Siten laite voisi tietää kalenteritietojen muuttumisesta ilman, että ne täytyy ladata kokonaan. Tällä tavoin on mahdollista säästää paljon energiaa etenkin isojen kalenterien tapauksessa. Kalenteritietojen koon selvittäminen onnistuu helposti HTTP-protokollan HEAD-kyselyn avulla, johon palvelin vastaa Content-Length-avainsanan avulla. Tämä menetelmä ei kuitenkaan ota huomioon niitä tilanteita, joissa kalenteritiedot muuttuvat, mutta niiden koko ei. Yksi esimerkki sellaisesta on tapahtuman ajankohdan siirtäminen. Tässä tapauksessa jokin ajankohta korvautuu toisella, eikä tietojen koko muutu.

4 TOTEUTUKSEN ANALYSOINTI

Laitteen toteutuksen kannalta on oleellista tarkastella erityisesti laitteen toiminta-aikaa ja valmistuskustannuksia. Laitteen toiminta-aika määrää lopulta sen, kuinka usein käyttäjä joutuu vaihtamaan paristoja tai kytkemään laitteen lataukseen. Valmistuskustannukset sen sijaan vaikuttavat tuotteen loppuhintaan, mikäli laitteesta tehdään myyntiin tarkoitettu tuote. Myyntivalmiin tuotteen toteuttaminen ei kuitenkaan ole tämän työn tavoite.

4.1 Lopullinen prototyyppi

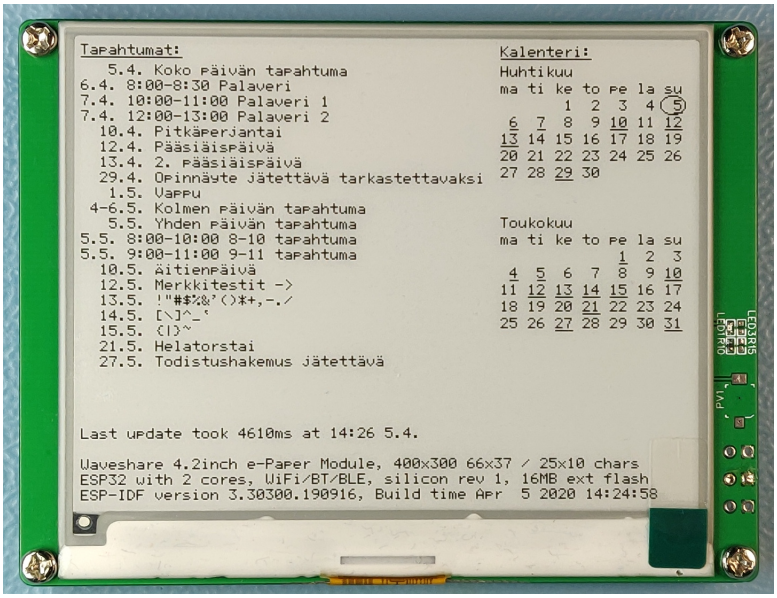
Lopullinen prototyyppi on esitetty kuvissa 4.1 ja 4.2. Kuvassa 4.2 nähdään, että laitteen etupuolella on itse näytön lisäksi ainoastaan oikeassa reunassa oleva merkkivalo. Kuvassa 4.1 näkyvät laitteen takapuolella sijaitsevan ESP32-piirin lisäksi ohjelmointiliitin, lineaariregulaattorin kytkentä, paikka hakkuriregulaattorille, näytön vaatima hakkurikytkentä, näytön liitin sekä painonapit ja usb-liitin. Lisäksi piirilevyllä on paikkoja komponenteille, jotka eivät ole oleellisia tämän työn kannalta. Tästä piirilevystä oli tarkoitus tehdä paranneltu versio, mutta sitä ei valitettavasti ehditty toteuttaa tämän työn puitteissa.



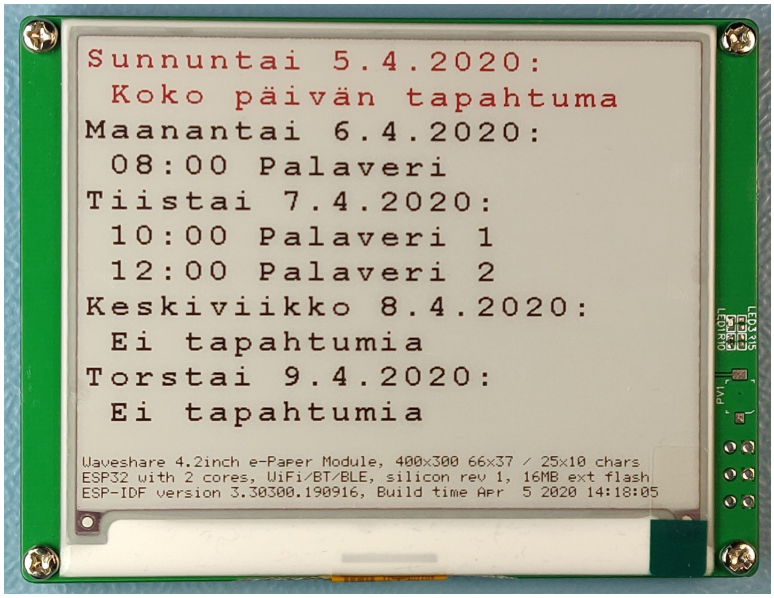
Kuva 4.1. *Kuva toteutetun laitteen piirilevystä.*

Kuvassa 4.3 on esitetty myös kolmivärisellä näytöllä varustettu versio. Tässä on käytetty myös isompaa kirjasintyyppiä, jolla on esitetty niin monen tulevan päivän tapahtumat kuin näytölle mahtuu. Tässä versiossa käytetty kirjasintyyppi ei välttämättä ole paras vaihtoehto

tekstin luettavuuden ja tiheyden suhteen, mutta parempaa kirjasintyyppiä ei löydetty tämän työn puitteissa.



Kuva 4.2. *Kuva toteutetun laitteen näytöstä, kun käytössä on pieni kirjasintyyppi.*



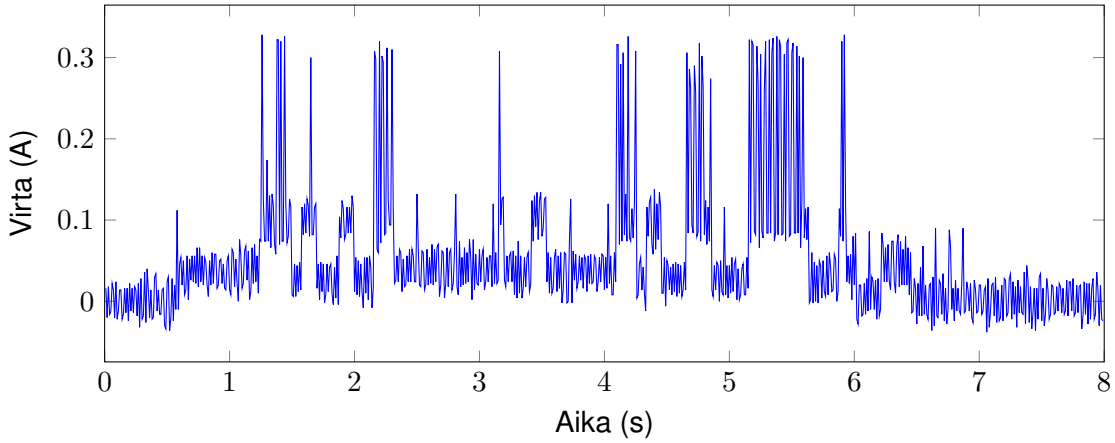
Kuva 4.3. *Kuva toteutetun laitteen näytöstä, kun käytössä on iso kirjasintyyppi.*

4.2 Laitteen toiminta-aika

Seuraavissa toiminta-aikalaskelmissa oletetaan, että laitteen energialähteellä on 2800 milliampeeritunnin sähkövaraus, kuten luvussa 3.1.2 on käsitelty. Lisäksi tässä ei oteta huomioon mitään energiamuunnosta, mikä mahdollistaa toiminta-ajan arvioinnin käyttäen energialähteen sähkövarausta ja laitteen keskikulutusta. Hakkuriregulaattorin käyttö parantaisi lievästi tätä arviota sen tekemästä energiamuunnoksesta johtuen. Lineaariregulaattorin käyttö sen sijaan heikentäisi arviota. Lisäksi lineaariregulaattoria ja paristoja käytet-

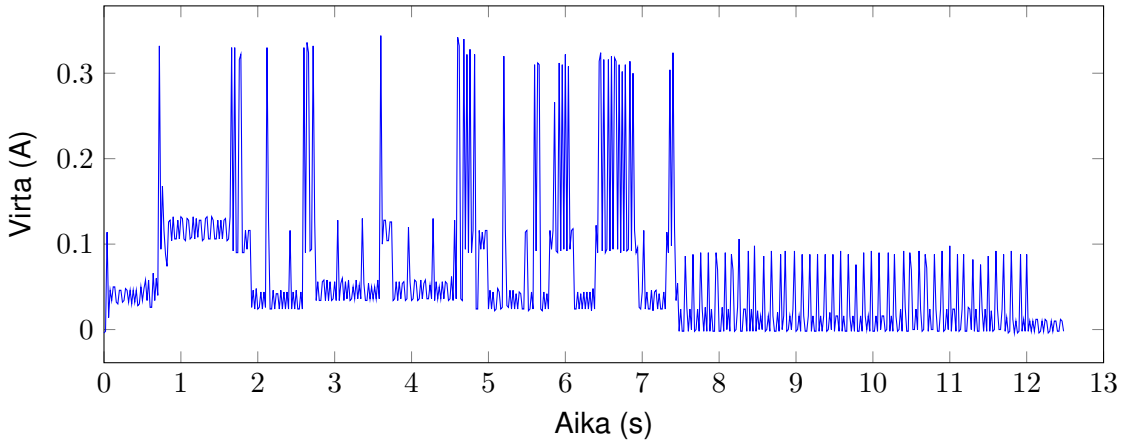
täessä laite lakkaisi toimimasta jo ennen paristojen tyhjentymistä, kuten luvussa 3.1.3 on käsitelty.

Laitteen virrankulutus on mitattu oskilloskoopilla käyttäen sarjavastusta. Oskilloskoopista otetun CSV-taulukon (Comma Separated Value) avulla voidaan myös laskea laitteen käyttämä sähkövaraus hakuprosessin aikana. Tämä onnistuu kertomalla kukin mittaustulos ajalla, joka kuluu mittausten välillä, ja laskemalla saadut luvut yhteen. Tässä ei oteta kantaa laitteen virrankulutukseen asetuksien muokkaustilassa, jolloin virrankulutus on todennäköisesti huomattavasti suurempi.



Kuva 4.4. *Optimoidun laitteen virrankulutus, kun näyttö päivitetään nopeasti.*

Kuvassa 4.4 on esitetty laitteen virrankulutus tietojen haun ja näytön päivittämisen aikana luvuissa 3.3.1 ja 3.3.2 käsiteltyjen optimointien kanssa. Tässä tapauksessa on käytetty kaksiväristä näyttöä ja laitteeseen on asetettu kolme kalenteria. CSV-taulukosta, josta kuva 4.4 on muodostettu, voidaan laskea edellä mainitulla tavalla, että laitteen käyttämä sähkövaraus on noin 0,43 ampeerisekuntia. Mikäli laitteen päivitysväli on 15 minuuttia, saadaan laitteen keskimääräiseksi virrankulutukseksi $0,43\,\mathrm{As}/(15 \cdot 60\,\mathrm{s}) = 477\,\mu\mathrm{A}$. Laitteen toiminta-ajaksi saadaan tällöin $2800\,\mathrm{mAh}/477\,\mu\mathrm{A} = 5870\,\mathrm{h}$, mikä tekee noin kahdeksan kuukautta. Mikäli laite rajoitetaan luvussa 3.3.3 esitetyllä tavalla toimimaan esimerkiksi klo 8 - 16, saadaan tämä toiminta-aika kolminkertaistettua kahteen vuoteen.



Kuva 4.5. *Optimoidun laitteen virrankulutus, kun näyttö päivitetään kokonaan.*

Tässä on kuitenkin otettava huomioon, että laitteen näyttö on välillä päivitettävä kokonaan. Laitteen virrankulutus tietojen haun aikana koko näytön päivityksen kanssa on esitetty kuvassa 4.5. CSV-taulukosta, josta kuva on muodostettu, voidaan jälleen laskea, että laitteen käyttämä sähkövaraus on tässä tapauksessa 0,79 ampeerisekuntia. Olettaen edelleen, että joka viides päivitys päivittää näytön kokonaan, laitteen käyttämä keskimääräinen sähkövaraus saadaan lausekkeesta $(0,43\,\mathrm{As} \cdot 4 + 0,79\,\mathrm{As})/5 = 0,5\,\mathrm{As}$. Tästä voidaan jälleen laskea laitteen keskimääräinen virrankulutus olettaen, että näyttö päivittyy 15 minuutin välein $0,5\,\mathrm{As}/(15 \cdot 60\,\mathrm{s}) = 556\,\mu\mathrm{A}$. Laitteen toiminta-aika saadaan jälleen kaavasta $2800\,\mathrm{mAh}/556\,\mu\mathrm{A} = 5036\,\mathrm{h}$, mikä tekee noin 7 kuukautta. Mikäli laitteen toimintaa rajoitetaan kuten yllä, voidaan tämä toiminta-aika kolminkertaistaa 21 kuukauteen.

4.3 Virrankulutus unitilassa

Lineaariregulaattorilla varustetun laitteen virrankulutukseksi mitattiin tässä työssä noin 43 mikroampeeria riippumatta siihen kytketystä näytöstä. Vastaavasti hakkuriregulaattorilla varustettu versio kuluttaa unitilassa noin 49 mikroampeeria, kun käyttöjännite on 4,5 volttia. Eri näyttöjen kanssa edellä mainituilla kytkennöillä oli noin 1 mikroampeerin ero virrankulutuksessa, mutta se johtuu todennäköisesti mittaustarkkuudesta. Lineaariregulaattorilla varustetun laitteen virrankulutus on peräti pienempi kuin mitä luvussa 3.1.6 on laskettu. Hakkuriregulaattorilla varustetun laitteen virrankulutus sen sijaan on vähän isompi kuin laskettu arvo. Erot lukemien välillä saattavat johtua mittaustarkkuudesta tai komponenttien yksilökohtaisista eroista.

Molempien regulaattoripiirien tapauksessa koko kytkennän virrankulutus on alle 50 mikroampeeria laitteen ollessa unitilassa. Olettaen energialähteen sähkövaraukseksi 2800 milliampeerituntia, kuten luvussa 3.1.2 on käsitelty, saadaan $2800\,\mathrm{mAh}/50\,\mu\mathrm{A} = 56\,000\,\mathrm{h}$. Tämä tarkoittaa yli kuuden vuoden toiminta-aikaa, mikäli laite olisi pelkästään unitilassa. Unitilan virrankulutus on otettava mukaan luvussa 4.2 laskettuun toiminta-aika-arvioon. Unitilan virrankulutus on sen verran pieni, että kokonaisvirrankulutusta voidaan arvioida yksinkertaisesti laskemalla unitilan ja aktiivitilan virrankulutukset yhteen lausekkeella $556\,\mu\mathrm{A} + 50\,\mu\mathrm{A} = 606\,\mu\mathrm{A}$. Tällä virrankulutuksella laitteen toiminta-ajaksi saadaan laskettua $2800\,\mathrm{mAh}/606\,\mu\mathrm{A} = 4620\,\mathrm{h}$, mikä tekee noin 6,5 kuukautta.

Mikäli laite asetetaan toimimaan vain osan ajasta kuten luvussa 3.3.3 on esitetty, saadaan toiminta-aika kolminkertaistettua noin 19 kuukauteen. Jos sen sijaan laitteen energialähteenä toimivat neljä AAA-paristoa, on laitteen toiminta-aika vain 57% verrattuna AA-paristoihin, kuten luvussa 3.1.2 on laskettu. Tällöin laitteen toiminta-ajaksi saadaan $19\,\mathrm{kk} \cdot 0,57\% = 10,83\,\mathrm{kk}$, eli noin 11 kuukautta.

4.4 Hinta

Toteutetun laitteen oleellisten komponenttien hinnat on esitetty taulukossa 4.1. Taulukosta nähdään, että lineaariregulaattorilla toteutetun laitteen kokonaishinta on 31 € ja

hakkuriregulaattorilla toteutettuna 36 €. Suurin osa laitteessa käytetyistä komponenteista on hankittu TME-kaupasta. Vain näyttö on hankittu Aliexpress-kaupasta, sillä TME:llä ei ollut myynnissä tähän työhön valittua näyttömallia. Vastaava eri valmistajan näyttö olisi kuitenkin maksanut 38 €, mikä olisi vaikuttanut lopputuotteen hintaan huomattavasti [5]. Piirilevy on hankittu JLCPCB-palvelusta, koska se on kätevästi integroitu työssä käytettyyn EasyEDA-työkaluun. Kytkennässä on luonnollisesti muitakin komponentteja, mutta niiden vaikutus kokonaiskustannuksiin on kovin pieni. Lisäksi on huomioitava, että kyseessä on vain arvio laitteen kustannuksista. Mikäli laitteesta tehtäisiin markkinoille sopiva laite, vaikuttaisivat muun muuassa tilausmäärät komponenttien lopulliseen hintaan.

Taulukko 4.1. *Laitteessa käytettyjen oleellisten komponenttien hinnat.*

Komponentti	Hinta (€)	Kauppa
ESP32	6	TME
EPD	20	Aliexpress
Lineaariregulaattori	3	TME
Hakkuriregulaattori	8	TME
Piirilevy	2	JLCPCB
Summa lineaariregulaattorilla	31	
Summa hakkuriregulaattorilla	36	

Tästä hintatarkastelusta on jätetty pois vähemmän kalliit komponentit sekä kotelo ja energiavarasto. Kotelon suunnittelu ei ollut tämän työn tavoitteissa ja energiavaraston hinta riippuu valinnasta paristojen ja akun väliltä. Sopiva kolmen AA-pariston paristokotelo kustantaisi alle 2 €, joten sen vaikutus kokonaishintaan olisi melko pieni [52]. Paristojen kapasiteettia vastaavan akun hinta olisi sen sijaan noin 15 €, mikä on huomattava kustannus koko laitteen hintaan nähden [53]. Tämän akun avulla laitteesta saataisiin kuitenkin paljon kompaktimpi, sillä akun paksuus on vain 7 millimetriä, kun taas paristokotelon paksuus on 17 millimetriä.

4.5 Hakkuriregulaattorin ongelmat

Hakkuripiirin Burst Mode -tila aiheutti tässä työssä kaksi ongelmaa. Ensimmäinen ongelma liittyy laitteen ohjelmointiin. Ohjelmointitilassa ESP32-piiri ei kykene siirtämään hakkuria normaaliin toimintatilaan, koska piirin normaali ohjelmisto ei ole tällöin käynnissä. Tämän takia hakkuripiiri piti asettaa normaaliin toimintatilaan hyppylangalla. Laitteen kehittämisen kannalta siinä pitäisi olla jonkinlainen kytkentä ohjelmointiliittimestä hakkuripiiriin, jotta kytkentä toimisi normaalisti.

Toinen ongelma liittyy laitteen normaaliin käyttöön. ESP32-piirin käynnistymisestä kestää noin 800 millisekuntia käyttäjän kirjoittaman ohjelman käynnistymiseen. Tämä tarkoittaa sitä, että nämä 800 millisekuntia hakkuri toimii vielä Burst Mode -tilassa. Käynnistyessään ESP32-piiri kuluttaa myös noin 60 milliampeeria, mikä on lähellä Burst Mode -tilan ylärajaa. Tämä johtaa siihen, että ESP32-piirin käyttöjännite laskee liian alas, jolloin jännitteen

valvontapiiri resetoi ESP32-piirin. Jännitteen valvontapiirin 240 millisekunnin viiveen jälkeen ESP32-piiri herää jälleen ja sama tilanne toistuu hyvin suurella todennäköisyydellä. Tässä työssä ongelma korjattiin kytkemällä diodi valvontapiirin ulostulosta hakkuripiirin MODE/SYNC-sisäänmenoon. Tällä tavoin ESP32-piirin resetoituminen aiheuttaa hakkuripiirin hetkittäisen siirtymisen normaaliin toimintatilaan, jonka aikana kytkennän ulostulokondensaattori ehtii latautua kokonaan. Tämän jälkeen ESP32-piiri ehtii käynnistyä ja siirtää hakkuripiirin normaaliin toimintatilaan, kuten sen on tarkoituskin.

Parempi ratkaisu tähän olisi sisällyttää kytkentään jokin toinen komponentti, joka voisi ensin siirtää hakkuripiirin normaaliin toimintatilaan ja vasta sitten herättää ESP32-piirin unitilasta. Todennäköisesti riittäisi myös tehdä nämä asiat samalla hetkellä. Tähän käyttötarkoitukseen sopisi esimerkiksi pieni mikrokontrolleri tai ajastinominaisuudella varustettu RTC-piiri. Tätä ei kuitenkaan testattu tämän työn puitteissa.

5 YHTEENVETO

Tämän työn perusteella luvussa 1 kuvatun laitteen tekeminen on mahdollista niin, että sille saadaan mielekäs toiminta-aika. Mittauksia tehdessä havaittiin kuitenkin, että verkon kuormituksella ja haettavan datan määrällä on huomattava vaikutus laitteen toiminta-aikaan. Tätä vaikutusta ei kuitenkaan tutkittu tarkemmin tämän työn puitteissa. Tämän vuoksi työssä lasketut toiminta-aika-arviot ovat optimistisia arvioita.

Työn tekemistä helpottivat huomattavasti piirilevylle sijoitetut testipisteet ja merkkivalo. Testipisteiden avulla laitteeseen oli helppo kiinnittää teholähde ja tarvittavat mittalaitteet. Merkkivalo sen sijaan auttoi havaitsemaan laitteen jumiutumisen sekä tapaukset, jossa laite ei siirtynytkään Light-sleep-unitilaan vaikka sen olisi pitänyt. Merkittävä parannus olisi myös lisätä piirilevylle selitteet, jotka kertoisivat, mihin kyseiset testipisteet on kytketty. Työssä oli myös kytketty nollan ohmin vastus regulaattorin ulostulon ja muun kytkennän väliin. Kyseinen vastus auttoi vianetsintää siinä vaiheessa, kuin toinen rakennetuista laitteista alkoi käyttäytyä virheellisesti. Regulaattorikytkennän pystyi tällä tavalla irrottamaan helposti lopusta kytkennästä irrottamalla vain yhden komponentin. Tämän vastuksen tilalla oli tarkoitus olla virranmittausvastus, mutta sitä ei lopulta käytetty.

5.1 Käytettävyys

Kuten luvussa 2.1.1 on mainittu, vaikuttaa laitteen käyttömukavuuteen näytön koon lisäksi myös siinä käytetty kirjasintyyppi. Kuvia 4.2 ja 4.3 vertaamalla voidaan nähdä, että pienempi kirjasintyyppi ei ole käyttökelpoinen kaukaa katsottuna. Isommalla kirjasintyypillä sen sijaan laitteeseen mahtuu liian vähän tietoa. Yksi ratkaisu tähän olisi käyttää isompaa näyttöä. Isompi näyttö olisi kuitenkin kalliimpi ja sen päivittäminen kuluttaisi mahdollisesti enemmän energiaa.

Laitteen päivitysvälinä on tässä työssä pidetty 15 minuuttia. Tämä on todennäköisesti riittävä, mutta on silti otettava huomioon mahdollisuus, että joku lähettää kalenterikutsun esimerkiksi vain 10 minuutin varoitusajalla. Tällainen tilanne on ihan mahdollinen työssäkäyvälle ihmiselle, jolloin päivitysväliä on lyhennettävä. Kotikäytössä sen sijaan päivitystahtia voi todennäköisesti pitää tätäkin pidempänä.

Tämän työn perusteella rakennettu laite sopii kaikkiin käyttötarkoituksiin, joissa tietoja pitää päivittää suhteellisen harvoin ja esitettävän tiedon määrä on suhteellisen pieni. Tämän rajauksen perusteella laite voisi näyttää esimerkiksi paikallissäätä, tehtävälistaa, tulevien sähköpostien otsikoita, uutisotsikoita, osakekursseja tai twitter-julkaisuja. Tämän työn

perusteella voidaan myös todeta, että laitteella voisi esittää enemmänkin informaatiota niin, että tietojen haun jälkeen laite tallettaisi muistiinsa enemmän tietoja kuin näytölle mahtuu. Tämän jälkeen käyttäjä voisi nappia painamalla lukea esimerkiksi lyhyen sähköpostin sisältöä. Tämä perustuu siihen, että näytön nopea päivittäminen vie paljon vähemmän energiaa kuin tietojen hakeminen. Tosin tällöin kuvanlaatu kärsii hieman.

5.2 Paristo- tai akkukäyttöisyys

Kolmea AA-paristoa käytettäessä laite voisi realistisesti toimia noin puolitoista vuotta ilman, että siitä joutuu vaihtamaan paristoja. Tämä nojaa siihen oletukseen, että laite päivittyy 15 minuutin välein ja vain osan päivästä. Se on mielestäni järkevä rajaus laitteen päivittämiseen ja myös järkevä väli paristojen vaihtamiseen.

Laitteen ohuen rakenteen takia laitteessa olisi mielekkäämpää käyttää AAA-paristoja, kuten luvussa 3.1.2 on ehdotettu. Tällöin olisi myös järkevää käyttää neljää paristoa, kunhan kytkennässä on käytössä sopiva hakkuriregulaattori. Tällä tavalla laitteeseen saataisiin edelleen noin 11 kuukauden toiminta-aika, kuten luvussa 4.3 on laskettu.

Akkukäyttöisenä laitteen voisi asettaa päivittymään useammin kuin 15 minuutin välein, mikäli jokin käyttötarkoitus sitä vaatii. Toki laitetta joutuu silloin lataamaan useammin, mutta se ei kuitenkaan tuo enempää lisäkustannuksia toisin kuin paristojen vaihto. Lisäksi luvussa 5.5 mainittava aurinkokenno voisi tehdä laitteesta sellaisen, että sitä ei tarvitse koskaan ladata. Tämä tekisi laitteesta aidosti langattoman ja paperista seinäkalenteria vastaavan tuotteen.

5.3 Regulaattorivalinnat

Työhön valittu lineaariregulaattori ei ollut hyvä valinta johtuen sen suhteellisen suuresta kynnysjännitteestä kuten luvussa 3.1.3 on mainittu. Parempi valinta laitteeseen olisi esimerkiksi XC6210-sarjan regulaattori, jonka kynnysjännite on huomattavasti pienempi [54]. Lisäksi kyseinen regulaattori olisi fyysisesti pienempi. Tämän työn pohjalta voidaan myös sanoa, että lineaariregulaattoria ei kannata käyttää, kun laitteen energiavarastona toimivat paristot. Akun kanssa lineaariregulaattori on sen sijaan ihan hyvä valinta, joskin hakkuriregulaattori tarjoaisi silti paremman hyötysuhteen.

Työhön valittu hakkuriregulaattori oli riittävän hyvä, vaikkakin se aiheutti ongelmia Burst mode -tilasta pois siirtymisen kanssa. Tämän työn perusteella laite kannattaisi kuitenkin toteuttaa niin, että kolmen pariston sijaan käytetään neljää. Työhön valittu hakkuriregulaattori ei kuitenkaan sovellu tähän käyttötarkoitukseen. Sen sijaan laitteessa voisi käyttää vaikkapa LTC1877-hakkuripiiriä, joka toimisi suuremmalla sisäänmenojännitteellä kuin käytetty LTC3440-piiri [55]. LTC1877-piirissä olisi myös huomattavasti pienempi tyhjäkäyntivirta, vaikka se ei olekaan tässä työssä rajoittava tekijä.

Yksi yleinen ongelma regulaattorien tapauksessa ainakin TME-kaupassa on se, että joitakin malleja ei saa ostettua yksittäiskappaleina vaan ainoastaan sadan tai tuhannen

kappaleen minimierissä. Tällaisen prototyypin rakentamiseen sellaisen määrän tilaaminen ei ole järkevää. Tässä tapauksessa olisi kuitenkin voinut käyttää mahdollisuutta tilata ilmaisia näytteitä suoraan valmistajalta. Esimerkiksi LTC1877-piirin valmistajalta Analog Devices -yritykseltä saisi tilattua ilmaisena näytteenä kaksi kappaletta kyseistä piiriä [56]. Tällaista mahdollisuutta ei kuitenkaan käytetty tässä työssä.

5.4 Laitteen ongelmat

Suurin ongelma tässä työssä toteutetussa laitteessa on se, että laitteen täytyy aina ladata kalenteritiedot kokonaisuudessaan. Tämä tarkoittaa sitä, että laitteen toiminta-aika riippuu vahvasti siitä, montako kalenteria laitteeseen on asetettu ja paljonko tapahtumia nämä sisältävät. Luvussa 3.3.3 on mainittu yksi tapa, jolla tältä voitaisiin välttyä, mutta se tekee laitteesta epävarman käyttää.

Toinen ongelma toteutetussa laitteessa on sen käytettävyys. Nykyaikaisissa laitteissa riittää, että syöttää Google-tilin käyttäjänimen ja salasanan, jotta pääsee tarkastelemaan kaikkia kyseiseen tiliin liittyviä kalentereita. Tässä työssä käytetyssä toimintatavassa on sen sijaan pakko syöttää erikseen jokaisen halutun kalenterin osoite. Tämä myös estää käyttämästä Google Calendar -palvelun sisäänrakennettuja kalentereita, esimerkiksi juhlapyhäkalenteria ja yhteystietojen syntymäpäivät -kalenteria, sillä näiden julkisia osoitteita ei yksinkertaisesti ole olemassa.

Molempien ongelmien ratkaisuna olisi käyttää työssä mainittua virallista Google Calendar -rajapintaa. Rajapinnan tapauksessa haluttuja kalenteritapahtumia voisi suodattaa palvelimella niin, ettei laitteen tarvitse vastaanottaa niin paljon dataa. On myös todennäköistä, että Google Calendar -rajapinnan kautta on mahdollista hakea palveluun tuotujen iCalendar-muotoisten kalenterien tapahtumat. Tällä tavalla laitteeseen saisi tuotua kaikki halutut kalenterit ilman, että jokaisen osoite täytyy kopioida erikseen. Haittapuolena tässä olisi se, että laitteen käyttämiseen tarvittaisiin välttämättä Google-tili, mutta ratkaisuna laite voisi tukea sekä iCalendar-formaattia että Google Calendar -rajapintaa.

5.5 Parannusehdotukset

Kuten luvussa 5.1 on esitetty, ei valittu näyttö ole riittävä tiedon esittämiseen tarpeeksi isolla kirjasintyypillä. Tämän takia laitteeseen sopisi paremmin isompi näyttö. Yksi vaihtoehto olisi myös käyttää sähkökirjojen lukulaitteissa käytettyjen näyttöjen varaosia. Näitä ei ole virallisesti saatavilla mistään, mutta esimerkiksi ED060SC7-näyttöjä on saatavilla Aliexpress-kaupassa runsaasti. Kyseisen näytön koko on 6 tuumaa, resoluutio on 800x600 ja hinta on noin 20 € [57]. Kyseinen näyttö olisi siis paljon parempi vaihtoehto hintaansa nähden. Tällaisen näytön käyttö on kuitenkin hyvin epävarmaa sen saatavuuden vuoksi.

Laitteen pienen virrankulutuksen vuoksi siihen sopisi myös erittäin hyvin pieni aurinkopaneeli. Siinä tapauksessa laitteen voisi toteuttaa pienemmällä akulla, joka latautuisi kyseisen aurinkopaneelin avulla. Tämä mahdollistaisi sellaisen laitteen rakentamisen, jota

ei tarvitse ladata koskaan. Aurinkopaneeli pitäisi kuitenkin mitoittaa niin, että se riittäisi lataamaan laitteen akkua sisätiloissa olevassa valaistuksessa. Kyseistä aurinkopaneelia voisi käyttää myös anturina laitteen päivittämistä varten. Laitteen tietoja ei nimittäin kannata päivittää, jos ympäristön valaistus ei mahdollista näytön lukemista. Tämän voisi toteuttaa myös tavallisella valoanturilla ilman aurinkopaneelia.

Laitteeseen saattaisi myös paremmin soveltua superkondensaattori akun sijaan. Superkondensaattori sopisi laitteeseen erittäin hyvin, mikäli laitteeseen lisättäisiin aurinkopaneeli. Tällöin ei tarvitsisi huolehtia litiumioniakun lataamisesta ja laitteesta saisi potentiaalisesti vieläkin kompaktimman.

Työssä jätettiin toteuttamatta toistuvien tapahtumien tuki. Tämän seurauksena laite näyttää toistuvista tapahtumista vain ensimmäisen esiintymisen. Toistuvat tapahtumat ovat melko oleellinen ominaisuus iCalendar-formaatissa ja laitteeseen olisi todennäköisesti suhteellisen helppo toteuttaa tuki tälle ominaisuudelle.

Laitteessa olisi myös edelleen varaa virrankulutuksen pienentämiseen. ESP32-piirin kellotaajuutta voi laskea, mikä toisaalta heikentää sen suorituskykyä. Laitteen olisi voinut myös toteuttaa yksiytimisellä ESP32-piirillä. Lisäksi ESP32-piiristä löytyy muitakin energiansäästöominaisuuksia, joita ei kuitenkaan tutkittu tässä työssä. Vastaavasti sähköpaperinäytön virrankulutusta olisi voinut tarkastella lähemmin. Tämä onnistuisi helpoiten niin, että ESP32-piirin tilalla olisi jokin huomattavasti vähemmän energiaa kuluttava piiri. Näillä asioilla olisi kuitenkin todennäköisesti melko pieni vaikutus laitteen kokonaiskulutukseen, sillä suurin energian kuluttaja on työssä käytetty langaton lähiverkko.

5.6 Käytetyt työkalut

Työssä käytetty Visual Studio Code -editori osoittautui erittäin hyväksi työkaluksi. Kyseisessä editorissa on maksullisiin tuotteisiin verrattavia ominaisuuksia, vaikka se on ilmainen. Siitä löytyy syntaksin värikorostuksen lisäksi tuki koodin automaattiselle täydennykselle sekä mahdollisuus hakea helposti kohdat, joissa jokin objekti on esitetty, määritelty tai käytetty. Kyseinen editori osoittautui paljon paremmaksi ja vakaammaksi työkaluksi kuin esimerkiksi Atom-editori [58], joka on myös ilmainen ja oli aikaisemmin PlatformIO-paketin oletuseditori. EasyEDA-työkalu osoittautui myös suhteellisen toimivaksi ottaen huomioon, että se on ilmainen ja selaimessa toimiva työkalu. Ainoa puute kyseisessä työkalussa on, että kytkentäkaavion komponenteilla ei ole komponentin arvolle varattua kenttää. Lisäksi palvelusta valmiiksi löytyvistä komponenteista kannattaa aina varmistaa, että niiden jalkakuviot (footprint) soveltuvat komponenttien käsinjuottamiseen.

ESP32-piirin dokumentaatio, ESP-IDF-kirjastot ja ESP32-piirin koodiesimerkit ovat myös kattavia ja laadukkaita verrattuna esimerkiksi Espressif Systemsin edelliseen tuotteeseen eli ESP8266-piiriin [59]. ESP32-piirin ja sitä käyttävien moduulien ominaisuuksista ja käytöstä löytyy runsaasti dokumentaatiota valmistajan sivuilta. ESP-IDF-kirjastot ja koodiesimerkit löytyvät sen sijaan kätevästi GitHub-palvelusta. ESP32-piirin dokumentaatio ei kuitenkaan ole vieläkään täydellinen. Esimerkiksi ESP32-piirin ADC-muuntimen

sisäänmenoimpedanssia ei ole virallisesti ilmoitettu missään, vaikka se on suhteellisen tärkeä yksityiskohta. Espressif Systemsin virallinen ESP-Prog-ohjelmointilaite osoittautui myös ihan toimivaksi laitteeksi, joskin on hieman kyseenalaista, että se myydään ilman koteloa.

Sähköpaperinäytön dokumentaatio ja esimerkit ovat sen sijaan hieman puutteelliset. Dokumentaatiossa on esimerkiksi mainittu Power off ja Deep sleep -tilat, mutta näiden eroja ei ole käsitelty muuten kuin virrankulutuksen osalta. Lisäksi dokumentaatiossa ei ole määritelty näytön reset sekvenssiä ja se ilmenee ainoastaan koodiesimerkistä. Koodiesimerkeissä sen sijaan on ristiriitoja dokumentaation kanssa. Koodiesimerkeissä esimerkiksi välitetään eräälle komennolle kaksi parametriä, vaikka dokumentaation mukaan parametreja pitäisi olla vain yksi. Toisessa kohdassa komennolle ei välitetä yhtään parametriä, vaikka dokumentaation mukaan pitäisi. Koodiesimerkkien löytyminen GitHub-palvelusta on kuitenkin hyvä asia.

LÄHTEET

[1] *Electric Imp. Dev Center*. 2020. URL: https://developer.electricimp.com/hardware/power_story (viitattu 23. 02. 2020).

[2] Fernández, M., Casanova, E. ja Alonso, I. Review of Display Technologies Focusing on Power Consumption. *Sustainability* 7.8 (elokuu 2015), 10854–10875. ISSN: 2071-1050. DOI: 10.3390/su70810854. URL: http://dx.doi.org/10.3390/su70810854.

[3] *Lighting device for non-emissive displays*. eng. Pat. AU8238101 (A), Hak.nro AU19820038101 20010827, Hak. Nokia Corporation, 15.04.2002. URL: https://worldwide.espacenet.com/publicationDetails/biblio?FT=D&date=20020415&DB=EPODOC&locale=&CC=AU&NR=8238101A&KC=A&ND=3 (viitattu 15. 04. 2020).

[4] *Transfer Multisort Elektronik. Raystar Optronics RX240128A-FGN*. 2020. URL: https://www.tme.eu/fi/details/rx240128a-fgn/graafiset-lcd-naytot/raystar-optronics/ (viitattu 23. 02. 2020).

[5] *Transfer Multisort Elektronik. Display Elektronik DEE 400300A2-W*. 2020. URL: https://www.tme.eu/fi/details/dee400300a2-w/e-paperi/display-elektronik/dee-400300a2-w/ (viitattu 23. 02. 2020).

[6] *Digi-Key Electronics. Sharp Microelectronics LS044Q7DH01*. 2020. URL: https://www.digikey.fi/product-detail/fi/sharp-microelectronics/LS044Q7DH01/425-2910-ND/5054070 (viitattu 23. 02. 2020).

[7] *Specification. 2.9"e-Paper*. 2017. URL: https://www.waveshare.com/w/upload/e/e6/2.9inch_e-Paper_Datasheet.pdf (viitattu 02. 05. 2020).

[8] *Specification. 4.2"e-Paper*. 2017. URL: https://www.waveshare.com/w/upload/6/6a/4.2inch-e-paper-specification.pdf (viitattu 21. 03. 2020).

[9] *Specification. 7.5"e-Paper*. 2017. URL: https://www.waveshare.com/w/upload/b/b6/7.5inch-e-paper-specification.pdf (viitattu 02. 05. 2020).

[10] Heikenfeld, J., Drzaic, P., Yeo, J.-S. ja Koch, T. Review Paper: A critical review of the present and future prospects for electronic paper. *Journal of the Society for Information Display* 19.2 (2011), 129–156. DOI: 10.1889/JSID19.2.129. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1889/JSID19.2.129. URL: https://onlinelibrary.wiley.com/doi/abs/10.1889/JSID19.2.129.

[11] *Youtube. E-paper hacking: fastest possible refresh rate*. 2017. URL: https://www.youtube.com/watch?v=MsbiO8EAsGw (viitattu 04. 04. 2020).

[12] Horowitz, P. ja Hill, W. *The Art of Electronics*. 3rd. USA: Cambridge University Press, 2015. ISBN: 0521809266.

[13] *ESP32 Series. Datasheet*. 2020. URL: https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf (viitattu 18. 02. 2020).

[14] *Espressif. Product Ordering Information*. 2020. URL: https://www.espressif.com/sites/default/files/documentation/espressif_products_ordering_information_en.pdf (viitattu 18. 05. 2020).

[15] *ESP-IDF. Programming Guide*. 2020. URL: https://docs.espressif.com/projects/esp-idf/en/latest/get-started/ (viitattu 19. 02. 2020).

[16] *Espressif. ECO and Workarounds for Bugs*. 2020. URL: https://www.espressif.com/sites/default/files/documentation/eco_and_workarounds_for_bugs_in_esp32_en.pdf (viitattu 18. 05. 2020).

[17] Barsukov, Y. *Battery power management for portable devices*. eng. Artech House power engineering series. Boston: Artech House, 2013. ISBN: 1-5231-1700-1.

[18] Reddy, T. B. ja Linden, D. *Linden's handbook of batteries*. eng. 4th ed. New York, N.Y: McGraw-Hill, 2011. ISBN: 1-282-91428-6.

[19] *Renesas. 3.3 Volt logic characteristics and applications*. 2019. URL: https://www.idt.com/eu/en/document/apn/124-33v-logic-characteristics (viitattu 29. 02. 2020).

[20] *ESP32-WROOM-32. Datasheet*. 2019. URL: https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32_datasheet_en.pdf (viitattu 18. 02. 2020).

[21] Coombs, C. F. *Printed circuits handbook*. eng. 6th ed. McGraw-Hill handbooks. New York: McGraw-Hill, 2008. ISBN: 1-281-15923-9.

[22] *ESP-IDF. Github repository*. 2020. URL: https://github.com/espressif/esp-idf (viitattu 19. 02. 2020).

[23] Gajski, D. D. *Embedded System Design Modeling, Synthesis and Verification*. eng. 1st ed. New York, NY: Springer US, 2009. ISBN: 1-4419-0504-9.

[24] Prasad, A. *802.11 WLANs and IP networking security, QoS, and mobility*. eng. Artech House universal personal communications series. Boston: Artech House, 2005. ISBN: 1-58053-790-1.

[25] Pajuranta, J.-P. *Langattoman verkon tietoturva yrityksessä*. fin. Tampere: Tampereen ammattikorkeakoulu, 2019. URL: http://urn.fi/URN:NBN:fi:amk-2019052712365 (viitattu 18. 05. 2020).

[26] *Informational. Simple Network Time Protocol (SNTP) Version 4 for IPv4, IPv6 and OSI*. 2006. URL: https://tools.ietf.org/html/rfc4330 (viitattu 08. 03. 2020).

[27] *Internet Standard. Standard for the Format of ARPA Text Messages*. 1982. URL: https://tools.ietf.org/html/rfc822 (viitattu 08. 03. 2020).

[28] *Draft Standard. Hypertext Transfer Protocol – HTTP/1.1*. 1999. URL: https://tools.ietf.org/html/rfc2616 (viitattu 08. 03. 2020).

[29] Loshin, P. *TCP/IP clearly explained*. eng. 4th ed. The Morgan Kaufmann Series in Networking. Amsterdam: Morgan Kaufmann Publishers, 2003. ISBN: 1-281-75497-8.

[30] Viega, J. *Network security with OpenSSL*. eng. First edition. Sebastopol, California: O'Reilly, 2002. ISBN: 0-596-51658-4.

[31] *Cloudflare. What Happens in a TLS Handshake?* 2020. URL: https://www.cloudflare.com/learning/ssl/what-happens-in-a-tls-handshake/ (viitattu 29. 03. 2020).

[32] *Google. Google Calendar API.* URL: https://developers.google.com/calendar (viitattu 02. 05. 2020).

[33] *Google. Google Apps Script.* URL: https://developers.google.com/apps-script (viitattu 02. 05. 2020).

[34] *WebCal.fi. Muistaa pyhäsi.* 2020. URL: http://www.webcal.fi/fi-FI/index.php (viitattu 29. 03. 2020).

[35] *Proposed Standard. Internet Calendaring and Scheduling Core Object Specification (iCalendar).* 2009. URL: https://tools.ietf.org/html/rfc5545 (viitattu 29. 03. 2020).

[36] *iCalendar.org.* URL: https://icalendar.org/ (viitattu 02. 05. 2020).

[37] *G Suite Admin Help. SSL connections overview.* 2020. URL: https://support.google.com/a/answer/100181?hl=en (viitattu 29. 03. 2020).

[38] *Proposed Standard. Upgrading to TLS Within HTTP/1.1.* 2000. URL: https://tools.ietf.org/html/rfc2817 (viitattu 29. 03. 2020).

[39] *curl. command line tool and library for transferring data with URLs.* URL: https://curl.haxx.se/ (viitattu 02. 05. 2020).

[40] *The World's Largest Web Developer Site. HTML Forms.* 2020. URL: https://www.w3schools.com/html/html_forms.asp (viitattu 08. 03. 2020).

[41] *EasyEDA. An Easier and Powerful Online PCB Design Tool.* 2019. URL: https://easyeda.com/ (viitattu 04. 04. 2020).

[42] *MIC5239. Low Quiescent Current 500mA uCap LDO Regulator.* 2007. URL: http://ww1.microchip.com/downloads/en/devicedoc/mic5239.pdf (viitattu 21. 03. 2020).

[43] *LTC3440. Micropower Synchronous Buck-Boost DC/DC Converter.* 2016. URL: https://www.analog.com/media/en/technical-documentation/data-sheets/3440fd.pdf (viitattu 21. 03. 2020).

[44] *ESP-Prog Board. Github repository.* 2019. URL: https://github.com/espressif/esp-iot-solution/blob/master/documents/evaluation_boards/ESP-Prog_guide_en.md (viitattu 21. 03. 2020).

[45] *PuTTY. a free SSH and Telnet client.* URL: https://www.chiark.greenend.org.uk/~sgtatham/putty/ (viitattu 02. 05. 2020).

[46] *APX809S/810S. 3-Pin Micro Power Voltage Detector.* 2019. URL: https://www.diodes.com/assets/Datasheets/APX809S-810S.pdf (viitattu 21. 03. 2020).

[47] *MCP73831/2. Miniature Single-Cell, Fully Integrated Li-Ion, Li-Polymer Charge Management Controllers.* 2014. URL: http://ww1.microchip.com/downloads/en/DeviceDoc/20001984g.pdf (viitattu 21. 03. 2020).

[48] *PlatformIO.* 2020. URL: https://platformio.org/ (viitattu 21. 03. 2020).

[49] *Visual Studio Code.* 2014. URL: https://code.visualstudio.com/ (viitattu 21. 03. 2020).

[50] *e-Paper. Github repository*. 2020. URL: https://github.com/waveshare/e-Paper (viitattu 21. 03. 2020).

[51] *Proposed Standard. Hypertext Transfer Protocol – HTTP/1.1*. 1997. URL: https://tools.ietf.org/html/rfc2068 (viitattu 07. 04. 2020).

[52] *Transfer Multisort Elektronik. 2465 KEYSTONE*. URL: https://www.tme.eu/fi/details/keys2465/paristot-kannattimet-ja-kahvat/keystone/2465/ (viitattu 03. 05. 2020).

[53] *Transfer Multisort Elektronik. LP674261 CELLEVIA BATTERIES*. URL: https://www.tme.eu/fi/details/accu-lp674261_cl/akut/cellevia-batteries/lp674261/ (viitattu 03. 05. 2020).

[54] *XC6210 Series. High Current, High Speed LDO Regulators*. URL: https://www.torexsemi.com/file/xc6210/XC6210.pdf (viitattu 06. 05. 2020).

[55] *LTC1877. High Efficiency, Monolithic Synchronous Step-Down Regulator*. URL: https://www.analog.com/media/en/technical-documentation/data-sheets/1877fb.pdf (viitattu 06. 05. 2020).

[56] *Analog Devices. Sample Products*. URL: https://www.analog.com/en/support/customer-service-resources/sales/sample-products.html (viitattu 06. 05. 2020).

[57] *AliExpress*. URL: https://www.aliexpress.com/ (viitattu 02. 05. 2020).

[58] *Atom. A hackable text editor for the 21st Century*. URL: https://atom.io/ (viitattu 06. 05. 2020).

[59] *ESP8266EX. Low-power, highly-integrated Wi-Fi solution*. URL: https://www.espressif.com/en/products/socs/esp8266ex/overview (viitattu 06. 05. 2020).