Tampereen yliopisto

Antti Jokinen

IOT LED-VALO ARDUINO-KEHITYSALUSTALLA

Tieto- ja sähkötekniikan tiedekunta
Kandidaatintyö
Maaliskuu 2019

TIIVISTELMÄ

ANTTI JOKINEN: IoT led -valo Arduino kehitysalustalla
Tampereen yliopisto
Kandidaatintyö, 45 sivua, 0 liitesivua
Maaliskuu 2019
Tieto- ja sähkötekniikan TkK-tutkinto-ohjelma
Pääaine: Elektroniikka
Tarkastaja: Yliopistonlehtori Erja Sipilä

Avainsanat: IoT, Arduino, led, MQTT, langaton lähiverkko

Tässä kandidaatintyössä on esitelty Arduino kehitysalustalla suunniteltu ja rakennettu IoT led -valo. IoT led -valon keskeiset komponentit ovat Arduino Nano, ESP8266 ESP-01S WLAN-moduuli, RGB led-nauha ja jännitelähde. Johdannossa esitetyt vaatimukset laitteen toiminnalle ovat seuraavat: led-nauha voidaan kytkeä päälle ja pois päältä sekä sen väriä voidaan muuttaa valmiin IoT-alustan kautta ja samat toimenpiteet voidaan suorittaa myös laitteen fyysiseltä käyttöliittymältä, johon kuuluu painonappi ja potentiometri. Lisäksi led-nauhan tulisi tuottaa tarpeeksi valoa, jotta se soveltuu tunnelmavalaistukseksi.

Led-nauhan ohjaamiseen käytetään PWM-himmennystä, jossa led-nauhan läpi kulkevaa virtaa hallitaan sen kanssa sarjaan kytketyillä transistoreilla, joiden kanava johtaa niiden hiloille syötetyn PWM-signaalin pulssisuhteen mukaan. Led-nauhan eri värit luodaan sen sisäisten eri väristen ledien kirkkauksien suhteina. IoT led -valo käyttää teholähteenään valmista hakkurijännitelähdettä. Kaikki laitteen komponentit on sijoitettu yhdelle yksipuoliselle piirilevylle, joka valmistettiin valotusmenetelmällä.

Arduino Nano on laitteen ohjausyksikkö, ja se ohjaa led-nauhaa sekä fyysiseltä käyttöliittymältä tulevien signaalien, että IoT-alustalta ESP:n kautta tulevien signaalien avulla. ESP pitää yllä yhteyttä IoT-alustaan, lähettää sinne Arduinolta tulevat led-nauhan tilan muutokset ja välittää IoT-alustalta tulevat led-nauhan ohjauskäskyt Arduinolle.

Laite käyttää IoT-alustan kanssa kommunikointiin MQTT-protokollaa, joka perustuu publish-subscribe -malliin. MQTT-palvelimeen yhteydessä olevat IoT-laitteet ilmoittavat palvelimelle, mistä datasta ovat kiinnostuneet ja palvelin lähettää kyseisen datan näille laitteille sen päivittyessä. MQTT-kommunikointi on toteutettu tässä laitteessa valmiiden kirjastojen avulla.

IoT led -valon tietoturvariskeihin kuuluu sen alttius fyysiselle peukaloinnille ja mahdollinen tietoliikenteen salakuuntelu. Fyysisen peukaloinnin todennäköisyys on kuitenkin pieni, tietoliikenne on salattu TLS-protokollalla ja palvelimen identiteetti varmennetaan.

IoT led -valon toteutus onnistui vaatimusten mukaisesti. Sitä voitaisiin jatkokehittää esimerkiksi hyötysuhdetta parantamalla, luomalla erilaisia valoefektejä tai integroimalla se kotiautomaatiojärjestelmään.

ALKUSANAT

Tähän kandidaatintyöhön vaadittava osaaminen on karttunut itsenäisen työn lisäksi monissa harjoitus- ja hupiprojekteissa, joten haluaisin kiittää opiskelukavereitani usein aamuyön tunneille venyneistä, joskus myös kärsivällisyyttä koettelevista, mutta aina vähintäänkin nautittavista tunkkaushetkistä.

Haluaisin myös kiitää Erja Sipilää kandidaatintyöni ohjauksesta ja palautteesta, jonka avulla sain työn tehtyä. Kiitokseni myös ihmisille, jotka väsymättä opastavat ja neuvovat meitä internetin elektroniikka- ja tietotekniikkapalstoilla.

Tampereella, 22.10.2019

Antti Jokinen

SISÄLLYSLUETTELO

LYHENTEET JA MERKINNÄT

ADC	Analog-To-Digital Converter, analogia-digitaalimuunnin
API	Application Programming Interface, ohjelmointirajapinta
EEPROM	Electronically Erasable Programmable Read-Only Memory, haihtumaton puolijohdemuisti
EMC	Electromegnetic Compatibility, sähkömagneettinen yhteensopivuus
ESP	ESP8266-01S WLAN-moduuli
HTTP	Hypertext Transfer Protocol, hypertekstin siirtoprotokolla
IDE	Integrated Development Environment, integroitu ohjelmointiympäristö
IoT	Internet of Things, esineiden internet
I/O	Input/Output, sisäänmeno/ulostulo
Led	Light-Emitting Diode, hohtodiodi
MOSFET	Metal-Oxide-Semiconductor Field-Effect Transistor, Metallioksidipuolijohdekanavatransistori
MQTT	Message Queuing Telemetry Transport, kevyt viestiprotokolla
PWM	Pulse-Width Modulation, pulssinleveysmodulaatio
QoS	Quality of Service, palvelutaso
RGB	Red, Green, Blue, punainen, sininen, vihreä
SHA	Secure Hash Algorithm, kryptografinen tiivistealgoritmi
SSL	Secure Socets Layer, kuljetuskerroksen suojaus
TCP/IP	Transmission Control Protocol/Internet Protocol, lähetysohjausprotokolla/internet protokolla
TLS	Transport Layer Security, kuljetuskerroksen suojaus
UART	Universal Asynchronous Receiver Transmitter, universaali asynkroninen lähetin-vastaanotin
USB	Universal Serial Bus, universaali sarjaväylä
UV	Ultraviolet, ultravioletti
WLAN	Wireless Local Area Network, langaton lähiverkko

1. JOHDANTO

Esineiden internet (engl. Internet of Things, IoT) on esineiden, eli erilaisten datayhteydellä varustettujen laitteiden kommunikointia keskenään ja käyttäjiensä kanssa halutun tehtävän suorittamiseksi. Esineet voivat olla esimerkiksi sensoreita, aktuaattoreita, älypuhelimia, tietokoneita jne. IoT on yleistymässä sekä teollisessa että kotitalouksien käytössä. Teollisuudessa sen hyötyjä voidaan käyttää esimerkiksi automaatiossa, logistiikassa, projektien hallinnassa ja teollisessa tuotannossa. Kotitaloussovelluksia voidaan puolestaan käyttää esimerkiksi kodin automaatioon, terveydenhoitoon ja älykkäisiin autoihin. [1, 2]

Tässä kandidaatintyössä tarkoituksena on toteuttaa IoT led-valo, jota voidaan ohjata sekä IoT-alusta Adafruit IO:n kautta internetselaimesta että fyysisellä kytkennällä. Tämän ohella on tarkoitus oppia ja selittää myös yleisemällä tasolla, miten tämän kaltaiset IoT-sovellukset toimivat, ja miten Arduino-alustaa voidaan hyödyntää niissä.

Laite koostuu Arduino Nanosta, ESP8266 langaton lähiverkko (engl. Wireles Local Area Network, WLAN) -moduulista ja hohtodiodi (engl. led, Light Emitting Diode) -valonauhasta. Led-nauhan tulisi tuottaa tarpeeksi valoa, jotta se soveltuu tunnelmavalaistukseksi. Valon voi kytkeä päälle tai pois päältä, ja sen väriä pystyy muuttamaan internetin välityksellä. Molemmat näistä operaatioista voidaan tehdä myös laitteen fyysisellä käyttöliittymällä. Laite toimii siis myös ilman verkkoyhteyttä, mutta pääasiallisesti sen oletetaan olevan yhteydessä internettiin langattoman lähiverkon kautta.

Tämä kyseinen laite on valittu toteutettavaksi, sillä se edustaa tyypillistä kodin automaation sovellusta, valaistusta, ja se on riittävän yksinkertainen toteuttaa rajallisilla alkutiedoilla. Syy IoT-tuen lisäämisessä valaistukseen on käyttäjäystävällisyydessä, valaistusta voidaan kontrolloida mistä tahansa laitteesta, jossa on internetselain, myös kodin ulkopuolelta esimerkiksi valaistuksen unohtuessa päälle.

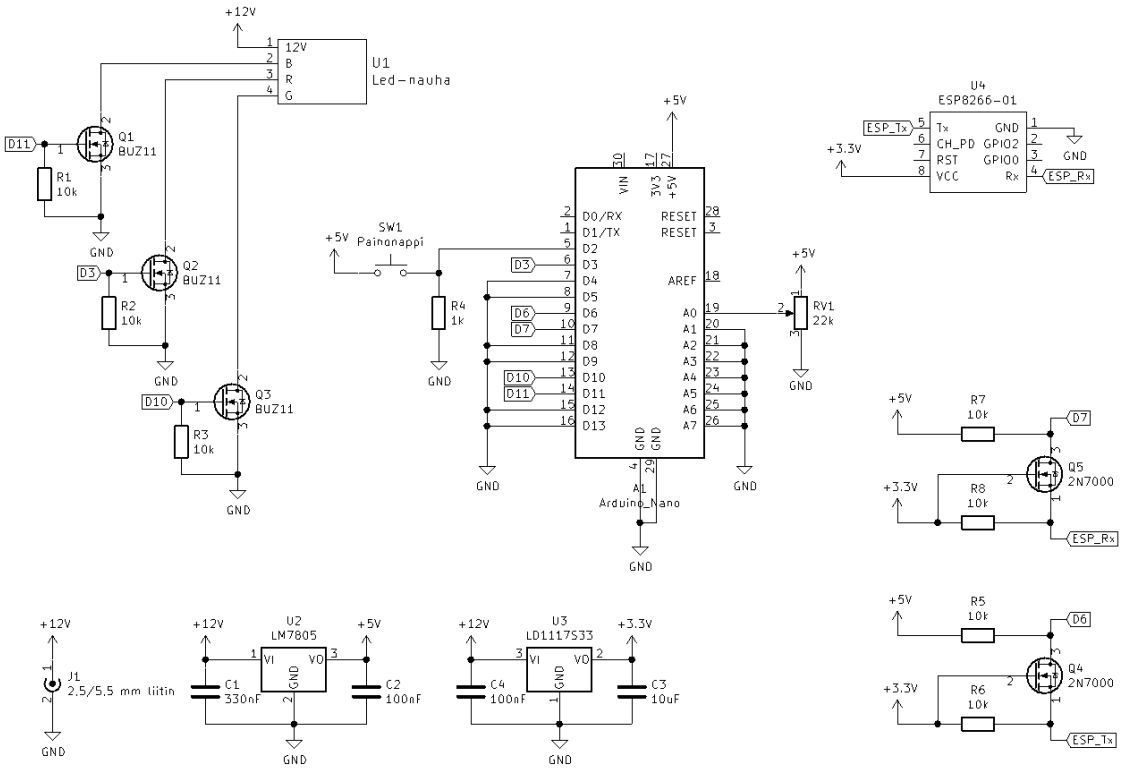
Luvun 1 johdannon jälkeen tämän työn luvussa 2 esitellään kytkennän toiminta ja lednauhan ohjaus, luvussa 3 on esitelty Arduinon toiminta, ja luvussa 4 esitetään WLAN-moduuli, sekä sen yhdistäminen internettiin. Luvussa 5 selostetaan laitteen kommunikaatio IoT-alustan kanssa, ja siihen liittyvät protokollat. Luvun 6 aihe on laitteen tietoturva, ja luku 7 on yhteenveto.

2. LED-VALON KYTKENTÄ

IoT led-valon piirisuunnittelussa ja komponenttivalinnoissa pyrittiin yksinkertaisuuteen ja edullisuuteen, kuitenkin niin, että laite täyttää johdannossa esitetyt vaatimukset. Kytkentäkaavio ja piirilevyn kytkentäkuvio suunniteltiin KiCad-ohjelmistolla.

2.1 Komponentit ja niiden kytkentä

Kuvassa 1 esitetty IoT led -valon kytkentä voidaan jakaa viiteen osaan. Vasemmassa yläkulmassa on led-nauhan ohjauskytkentä, keskellä Arduinon kytkentä fyysiseen käyttöliittymään, oikeassa yläkulmassa WLAN-moduulin kytkentä, vasemmassa alakulmassa teholähteen ja jänniteregulaattorien kytkennät sekä oikeassa alakulmassa logiikkatason muuntajien kytkentä.



Kuva 1. *IoT led -valon kytkentäkaavio*

Led-nauhan käyttöjännite on 12 V, joten mahdollisimman yksinkertaisen kytkennän aikaansaamiseksi se valittiin tämän laitteen käyttöjännitteeksi. Arduinon ja WLAN-moduulin tarvitsemat alhaisemmat käyttöjännitteet saadaan aikaan lineaarisilla jänniteregulaattoreilla L7805CV ja LD1117V33C. Molemmat tuottavat käyttötarkoitukseensa tarpeeksi tarkan ulostulojännitteen ja kestävät 12 V sisäänmenojännitteen, sekä virran, jonka piirit

tarvitsevat lämpenemättä liikaa [3, 4]. Jänniteregulaattorien lämpeneminen voidaan laskea tehohäviöiden ja lämpöresistanssin arvojen avulla. Ensin lasketaan regulaattorien tehohäviö eli lämmöksi muuttunut teho:

$$P = U * I = (U_I - U_O) * I, \quad (1)$$

missä P on tehohäviö, U on regulaattorin jännitehäviö eli sisäänmenojännitteen (U_I) ja ulostulojännitteen (U_O) erotus ja I on regulaattorin läpi kulkeva virta. Lämpöresistanssi R_θ on määritelty seuraavasti [5]:

$$R_\theta = \frac{\Delta T}{P} = \frac{T_C - T_A}{P}, \quad (2)$$

missä ΔT on lämpötilaero ympäröivän ilman (T_A) ja komponentin kotelon pinnan (T_C) välillä. Regulaattorien datalehdillä on annettu arvot lämpöresistanssille komponentin ja ympäröivän ilman välillä ($R_{\theta(ja)}$), joka molemmilla regulaattoreilla (TO-220-kotelo) on 50 C°/W [3, 4]. Kaavasta 2 voidaan johtaa regulaattorin kotelon lämpötila:

$$T_C = P * R_{\theta(ja)} + T_A \quad (3)$$

$$T_C = (U_I - U_O) * I * R_{\theta(ja)} + T_A \quad (4)$$

Arduinon keskimääräiseksi virrankulutukseksi, eli myös sen regulaattorin läpi kulkevaksi virraksi, mitattiin noin 50 mA ja WLAN-moduulin keskimääräiseksi virrankulutukseksi noin 70 mA. Ilman lämpötila T_A oletetaan huoneen lämpötilaksi 25 ºC. Nyt molempien regulaattorien kotelon lämpötila voidaan laskea kaavasta 4:

$$T_{L7805} = (12\,V - 5\,V) * 0{,}050\,A * 50\frac{°C}{W} + 25\,°C \approx 42{,}5\,°C$$

$$T_{LD117} = (12\,V - 3{,}3\,V) * 0{,}056\,A * 50\frac{°C}{W} + 25\,°C \approx 55{,}5\,°C$$

Molemmat lämpötilat ovat paljon pienempiä kuin regulaattoreiden maksimitoimintalämpötila 125 ºC, joten regulaattorit eivät tarvitse ulkoista jäähdytyselementtiä. Jänniteregulaattorien sisään- ja ulostulokondensaattorit C1 - C4 on mitoitettu datalehtien ehdotuksien mukaan. [3, 4]

Koska WLAN-moduulin käyttöjännite on 3,3 V ja Arduinon 5 V, täytyy Arduinon vastaanottamia signaaleja vahvistaa, ja sen lähettämiä signaaleja vaimentaa, jotta Arduino lukee sisäänmenonsa oikein ja WLAN-moduuli ei hajoa liian suuresta I/O (engl. Input/Output, sisäänmeno/ulostulo) -pinnin jännitteestä [6, 7]. Tämä toteutetaan kahdella kaksisuuntaisella logiikkatason muuntajalla, jotka koostuvat transistoreista Q4 ja Q5 sekä vastuksista R5 – R8. ESP:n (ESP8266-01S WLAN-moduuli) sarjaliikennepinnit kytketään muuntajien 3,3 V puolelle ja Arduinon virtuaaliset sarjaliikennepinnit D6 ja D7 5 V

puolelle. Kun 3,3 V puoli lähettää loogisen signaalin 1, metallioksidi-puolijohdekanavatransistorin (engl. Metal-Oxide-Semiconductor Field-Effect Transistor, MOSFET) hilan ja lähteen jännite-ero on 0 V, sillä lähteen jännite on 3,3 V, mistä johtuen transistorin kanava on kiinni, jolloin sen nielulle kytketty vastus vetää 5 V puolen jännitteen 5 V:in. Vastaavasti 3,3 V puolen lähettäessä loogisen signaalin 0, transistorin hilan ja lähteen välinen jännite on 3,3 V, mikä avaa sen kanavan ja vetää 5 V puolen jännitteen samaksi kuin 3,3 V puolella, eli 0 V:in. 5 V puolen lähettäessä loogisen signaalin 1, transistorin kanava on kiinni ja sen lähteelle kytketty vastus vetää 3,3 V puolen 3,3 V:in. Kun 5 V puolelta lähetetty looginen signaali on 0, MOSFET:in substraattidiodi johtaa, sillä nielun jännite (0 V) on pienempi kuin lähteen (3,3 V). Tämä vetää lähteen jännitteen alas diodin kynnysjännitteeseen, eli noin 0,7 V:in, jolloin transistorin kanava aukeaa ja vetää 3,3 V puolen edelleen 0 V:in. [8]

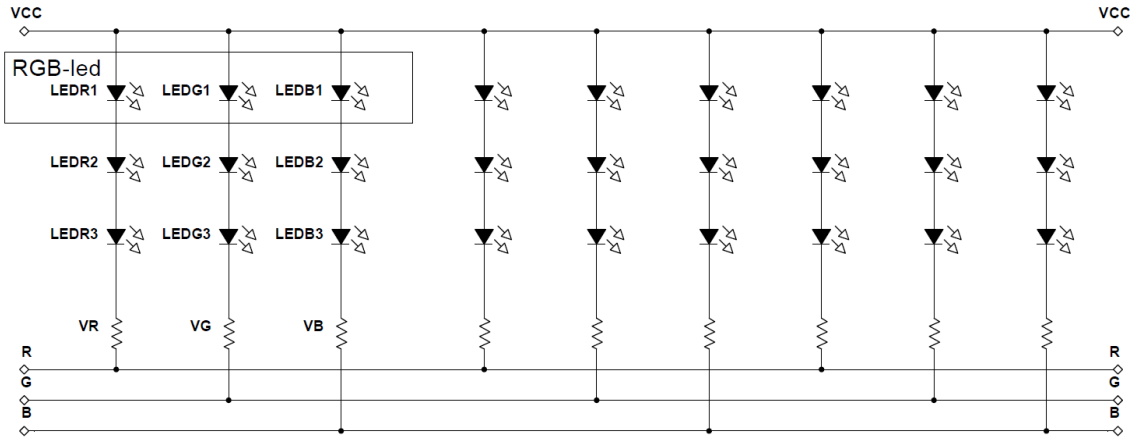
Vastukset R1 – R3 ovat alasvetovastuksia, jotka estävät transistorien Q1 - Q3 hilojen jännitteiden kellumisen. Vastus R4 on painonapin alasvientivastus, joka rajoittaa painonapin läpi kulkevaa virtaa painonapin ollessa painettuna ja vetää D3-pinnin jännitteen maatasoon kun painonappi ei ole painettuna. Teholähde kytketään kuvan 1 J1 liittimeen 2,5/5,5 mm DC-liittimellä.

Tämän laitteen fyysiseen käyttöliittymään kuuluu painonappi, jota painamalla led-nauhan tila vaihtuu, ja säädin, jota kääntämällä led-nauhan väriä voidaan muuttaa. Jotta led-nauhan väriä voidaan muuttaa sekä IoT-alustalta että säätimestä, säädin vaikuttaa väriin vain sitä kierrettäessä. Säädin toteutetaan potentiometrinä (RV1 kuvassa 1), joka kytketään toisesta päästään maahan, toisesta 5 voltin jännitteeseen ja sen keskimmäinen liitin kytketään Arduinon analogiseen sisäänmenoon A0. Potentiometrin säädintä kääntämällä saadaan aikaan jännitteenjako 0 voltin ja 5 voltin välillä, jonka tulos muutetaan Arduinon analogia-digitaalimuuntimen (engl. Analog-To-Digital converter, ADC) avulla kokonaisluvuksi väliltä 0 - 1023, joka puolestaan tulkitaan väriksi led-nauhan ohjausta varten.

2.2 Led-nauha ja sen ohjaustransistorit

Led-nauha on joustava piirilevy, joka koostuu osioista, joissa on kolme RGB (engl. Red Green Blue, sininen punainen vihreä) -lediä, sekä niiden etuvastukset. Jokaisen osion molemmissa päissä ovat liitosalueet, joten led-nauhaa voidaan lyhentää osio kerrallaan haluttuun mittaan. Yksittäinen RGB-led sisältää punaisen, vihreän sekä sinisen ledin, ja sen kyky tuottaa useita värejä perustuu näiden lähekkäin sijoitettujen yksiväristen ledien kirkkauksien suhteeseen, jolloin kauempaa katsottuna ne muodostavat halutun värin. Esimerkiksi valkoista valoa tuotettaessa asetetaan kaikkien kolmen ledin kirkkaus samaan tasoon, oranssiin valoon taas käytetään vain punaista ja vihreää lediä, joiden kirkkaus on lähes sama, ja niin edelleen. Kaikkien yksiväristen ledien toiset päät on kytketty yhteen, ja toiset päät kytketty saman väristen ledien kanssa yhteen, jolloin kunkin värisiä ledejä voidaan säätää erikseen niiden läpi kulkevaa virtaa muuttamalla. [9] Led-nauha voi olla

yhteisanodikytkentäinen, jolloin kaikkien ledien anodit on kytketty yhteen, tai yhteiskatodikytkentäinen, jossa kaikkien ledien katodit on kytketty yhteen. Kuvassa 2 on havainnollistettu led-nauhan ja RGB-ledin rakennetta kytkentäkaaviolla. Numeroidut ledit kuuluvat yhteen osioon. VCC on led-nauhan käyttöjännite ja kanavilla R, G ja B ohjataan RGB-ledien läpi kulkevaa virtaa, ja siten niiden väriä. Led-nauhan virtaa hallitaan passiivisesti vastusten VR, VG. ja VB avulla, joiden yli oleva jännite saadaan vähentämällä led-sarjan kolmen ledin kynnysjännitteet käyttöjännitteestä. Led-nauhan vastukset on mitoitettu niin, että tällä jännitteellä niiden ja siten myös ledien läpi kulkeva virta on sellainen, että ledit palavat, mutta eivät rikkoudu. [10]



Kuva 2. *Yhteisanodikytkentäisen RGB-led-nauhan kolmen osion kytkentäkaavio [11]*

Koska led-nauhan eri värit luodaan punaisten, vihreiden ja sinisten ledien kirkkauksien suhteena, on näitä kirkkauksia säädettävä himmentämällä ledejä, eli laskemalla niiden kirkkautta hallitusti. Koska Arduinolla voidaan tuottaa suoraan pulssinleveysmoduloituja signaaleja (engl. Pulse-Width Modulation, PWM), valitaan ledien himmennysmetodiksi PWM-himmennys analogisen himmennyksen sijaan. Analogisella himmennyksellä tarkoitetaan ledien kirkkauden säätämistä niiden DC-virtaa säätelemällä. PWM-himmennyksessä puolestaan säädetään ledien läpi kulkevan vakiovirran pulssisuhdetta, ja siten virran keskiarvoa. [12]

Led-nauhan himmennys suoritetaan kolmen MOSFET:in avulla. Himmennyksessä käytettävä transistorimalli on N-tyypin BUZ11. Tämä malli on valittu sen alhaisen noin 3 voltin kynnysjännitteen, kanavan virrankeston ja alhaisen kanavaresistanssin perusteella. Alhainen kanavaresistanssi on tärkeä parametri, sillä se määrittää transistorin tehohäviön suuruuden, joka puolestaan vaikuttaa sen lämpenemiseen. Tehohäviö lasketaan kaavalla:

$$P = {I_D}^2 * R_{DS(on)}, \qquad (1)$$

missä P on tehohäviö, I_D on transistorin kanavavirta ja $R_{DS(on)}$ on sen kanavaresistanssi. Tehohäviön avulla voidaan johtaa transistorin kotelon lämpötila kaavasta 3:

$$T_C = {I_D}^2 * R_{DS(on)} * R_{\theta(ja)} + T_A \quad (4)$$

I_D:n arvo on led-nauhan suurin yhden värikanavan läpi kulkeva virta, eli punaiselta kanavalta 12 V käyttöjännitteellä mitattu 825 mA. Kanavaresistanssin arvo 0,03 Ω ja lämpöresistanssin arvo 62,5 ºC/W saadaan BUZ11:ta datalehdeltä. Nyt voidaan laskea kotelon lämpötila:

$$T_C = (0{,}825\ A)^2 * 0{,}03\ \Omega * 62{,}5\frac{C°}{W} + 25\ °C \approx 26{,}3\ C°,$$

joka on lähes huoneen lämpötila, mikä tarkoittaa, että transistorit eivät tarvitse erillistä jäähdytyselementtiä.

Jokaisen transistorin kanava kytketään sarjaan yhden värikanavan (kuvassa 1 R, G ja B) kanssa, ja niiden hilat kytketään kukin yhteen Arduinon I/O-pinniin. Nyt RGB-ledien väriä voidaan ohjata Arduinosta transistorien hiloille kulkevalla PWM-signaalilla. [13]

2.3 Teholähteen valinta

Led-nauhan vaatiman tehon vuoksi paras vaihtoehto on käyttää verkkovirtaa esimerkiksi akkujen sijasta. Ledien teholähteen valinta vaikuttaa niiden kykyyn tuottaa valoa yhtenäisellä intensiteetillä, sekä niiden hyötysuhteeseen. Teholähde valitaan yksittäisten ledien kuluttaman tehon ja ledien kytkennän perusteella. Mikäli useita ledejä kytketään sarjaan, on paras vaihtoehto virtalähde, sillä silloin jokaisen ledin läpi kulkee sama virta, ja siten niiden tuottaman valon intensiteetti on mahdollisimman yhtenäinen. Jos taas ledit ovat korkeatehoisia, hakkurivirtalähde on niiden hyötysuhteen kannalta paras vaihtoehto. [10]

Led-nauha kuitenkin koostuu useista rinnankytketyistä kolmen ledin sarjoista, ja tällaisessa tapauksessa ideaalisin vaihtoehto olisi jännitelähde, ja jokaisen led-sarjan kanssa sarjaankytketty lineaarinen virtaregulaattori. Tällöin jokaisen led-sarjan läpi kulkisi sama virta, ja valon intensiteetti sarjojen kesken pysyisi mahdollisimman yhtenäisenä. Tätä kutsutaan aktiiviseksi virranhallitsemiseksi. Led-nauhan sisäiseen rakenteeseen ei kuitenkaan voida vaikuttaa, ja led-nauhaa halutaan tässä laitteessa esteettisistä syistä käyttää, joten yksittäisten led-sarjojen virtaa hallitaan vain passiivisesti led-nauhan sisäisten vastusten avulla. Ledien suuren määrän huomioiden tämä on virtaregulaattoreita edullisempi vaihtoehto, vaikkakin ledien hyötysuhteen kannalta huonompi, sillä vastusten jännitehäviöiden on oltava suuria, jotta virta pysyy halutuissa rajoissa. Vastukset taas muuttavat käyttämänsä tehon lämmöksi eli se menee hukkaan, mikä näkyy ledien hyötysuhteen alenemisena ja led-nauhan lämpenemisenä. Toinen passiivisen virranhallinnan heikkous on sen vaatimus stabiilille jännitelähteelle. Vaihtelu ledien käyttöjännitteessä saattaa aiheuttaa suurta ledien virran muutosta, joka puolestaan näkyy led-nauhan kirkkauden vaihteluina. Tämä johtuu yksittäisten ledien kynnysjännitteiden eroista, jotka yhdessä käyttö-

jännitteen vaihteluiden kanssa määrittävät led-sarjojen virranhallintavastusten jännitehäviön ja siten niiden virran. Mikäli kynnysjännite-erot ja käyttöjännitteen vaihtelu ovat siis suuria, niin myös virran vaihtelu led-sarjojen välillä on suurta. [10]

Led-nauhan tarvitsema teho voidaan laskea sen yhden osion virran perusteella. Led-nauhan värikanavien virroiksi 12 V jännitteellä mitattiin 825 mA punaiselle kanavalle, 805 mA vihreälle kanavalle ja 670 mA siniselle kanavalle. Led-nauhan yhteenlaskettu virrankulutus on siis 2,3 A. Lisäksi Arduinon ja sen jänniteregulaattorin maksimivirrankulutukseksi mitattiin 50 mA sekä WLAN-moduulin ja sen jänniteregulaattorin maksimi virrankulutukseksi 150 mA. Kokonaisvirrankulutus on siis suurimmillaan noin 2,5 A. Maksimitehonkulutus tällöin on noin 2,5 A * 12 V = 30 W.

Näiden reunaehtojen perusteella tähän laitteeseen on valittu CPS-12030C8-hakkurijännitelähde kiinteällä 12 V lähdöllä. Kuten taulukosta 1 nähdään, sen maksimiulostulovirta ja -teho ovat suurempia kuin laitteelle lasketut maksimivirrankulutus ja maksimitehonkulutus.

Taulukko 1. *CPS-12030C8-jännitelähteen keskeiset ominaisuudet [14]*

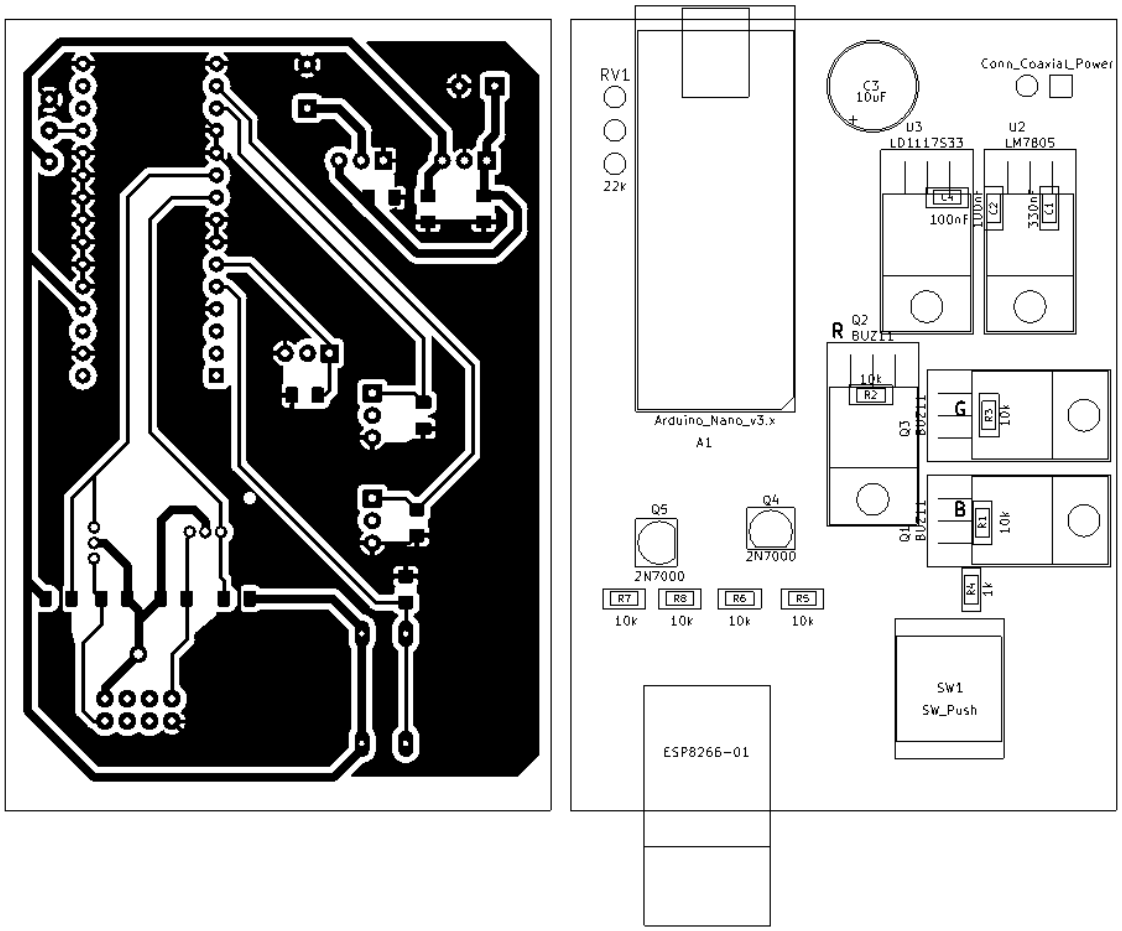
Ominaisuus	**Arvo**
Sisäänmenojännite	90 - 264 Vac
Ulostulojännite	12 Vdc
Maksimiulostulovirta	3 A
Maksimiulostuloteho	36 W
Maksimirippelijännite	1-2 %
Load regulation	± 5,0 %
Suojaukset	Ylijännite- ja virta, oikosulku
Käyttölämpötila	0 - 40 ºC

Laitteeseen on valittu valmis teholähde, sillä vaikka itse rakennettu teholähde olisi käyttötarkoitukseen erikoistuneempi, sen vaatima työmäärä olisi liian suuri. Lisäksi valmis teholähde täyttää vaaditut sähköturvallisuus- ja sähkömagneettinen yhteensopivuus (engl. Electromagnetic Combatibility, EMC) -standardit, eikä niistä tarvitse siten teholähteen osalta huolehtia.

2.4 Piirilevy ja sen valmistus

IoT led -valon piirilevy valmistettiin valotusmenetelmällä, jossa ensiksi piirilevyn kytkentäkuvio tulostetaan kalvolle. Tätä kutsutaan valotusmaskiksi. Seuraavaksi piirilevy, jonka pinnassa on ultravioletti (engl. Ultraviolet, UV) -säteilyyn reagoivaa lakkaa kuparikerroksen päällä, asetetaan valotusmaskin alle ja piirilevy altistetaan UV-säteilylle. Valotusmaskin ulkopuolisiin osiin osuva UV-säteily kehittää lakan, jolloin se voidaan poistaa kemiallisesti valotuksen jälkeen. Kun lakka on poistettu kehitetyiltä alueilta, voidaan kupari syövyttää näiltä samoilta alueilta. Kemiallisessa syövytysprosessissa valotusmaskin peittoon jäänyt lakka suojaa sen alla olevaa kuparia, jolloin se syöpyy vain halutuilta

alueilta. Lopuksi jäljelle jäänyt lakka liuotetaan kuparin päältä, jotta komponentit voidaan juottaa piirilevylle. IoT led -valon piirilevyn valotusmaski on esitetty kuvassa 3 vasemmalla. [15]

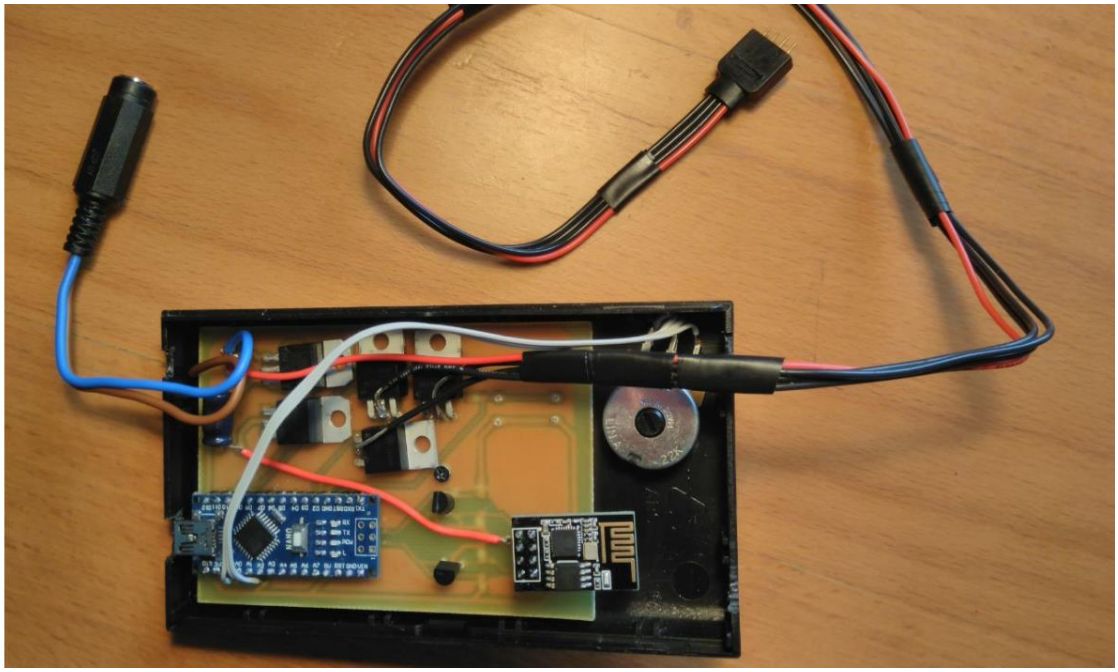


Kuva 3. *IoT led -valon piirilevyn valotusmaski ja komponenttien asettelu.*

Kuvan 3 valotusmaskin vetojen leveydet ovat 0,4 mm signaaleille ja 1,0 mm käyttöjännitteille. Kuparitäytön väli vetoihin on 0,7 mm. Piirilevy on toteutettu yksipuolisena, koska tarvetta kaksipuoliselle piirilevylle ei ollut ja vain yksi veto, 3,3 V ESP:lle, jouduttiin toteuttamaan hyppylangalla. Led-nauhan värikanavat on juotettu suoraan transistorien Q1 – Q3 nieluille. Käytetyt komponentit ovat läpiladottavia 100 nF ja 330 nF kondensaattoreita sekä vastuksia lukuun ottamatta, jotka ovat 1206-koteloisia pintaliitoskomponentteja.

Komponentit on aseteltu, kuvassa 3 oikealla, toiminnallisiin ryhmiin. Piirilevyn oikeassa yläkulmassa sijaitsevat 12 V sisäänmenon sekä jänniteregulaattoreiden kytkennät, niiden alapuolella led-nauhan ohjaustransistorit ja alimpana painonappi. Vasemmassa yläkulmassa on Arduino ja potentiometriin kulkevat johtimet, joiden alapuolella on logiikkatason muuntaja sekä ESP. ESP on kiinnitetty piirilevyyn 2x4-pinnisellä liittimellä, jotta sen voi helposti irrottaa ja ohjelmoida uudelleen tarvittaessa. Ohjelmointi ei ole mahdollista

ESP:n ollessa kytkettynä piirilevyyn. ESP:n asettelu siten, että sen piirilevyantenni sijoittuu piirilevyn ulkopuolelle, on toteutettu sen datalehden ohjeen mukaisesti [7].



Kuva 4. *Kalustettu piirilevy kiinnitettynä koteloon*

Kuvassa 4 on esitetty valmis IoT led- valo ilman kotelon takakantta, led-nauhaa ja jännitelähdettä. Kuvassa 4 vasemman puoleinen liitin on jännitelähteeseen kytkettävä 2,5/5,5 mm DC-liitin ja oikeanpuoleinen liitin on led-nauhaan kytkettävä 4-piikkinen piikkirima. Led-nauhan käyttöjännite on kytketty punaisella ja sen värikanavat mustilla johtimilla.

3. ARDUINO LAITTEEN OHJAUSYKSIKKÖNÄ

Arduino on helppokäyttöinen avoimen lähdekoodin sulautettujen järjestelmien kehitysalusta, joka on laajasti käytössä etenkin elektroniikkaharrastajien keskuudessa. Arduino-ympäristöön kuuluu erilaisia piirejä sekä ohjelmointiympäristö Arduino integroitu ohjelmointiympäristö (engl. Integrated Development Environment, IDE), jonka avulla piirejä ohjelmoidaan Arduino-ohjelmointikieltä käyttäen. Tässä työssä on käytössä Arduino Nano -kehitysalusta, jossa on ATmega328P-mikrokontrolleri, johon sen toiminnallisuudet perustuvat. [16]

Sulautettujen järjestelmien kehitysalustat ovat prototyyppien tai yksittäisten laitteiden suunnitteluun ja toteuttamiseen tarkoitettuja valmiita piirejä, jotka sisältävät mikrokontrollerin tai -prosessorin lisäksi sen yleisesti tarvitsemia oheiskomponentteja, kuten esimerkiksi jänniteregulaattoreita, kiteen tai USB (engl. Universal Serial Bus, universaali sarjaliikenneväylä) -liitynnän, sekä liityntöjä kontrollerin jalkoihin. Kehitysalustojen etu prototyyppien kehittämisessä on niiden edullisuus, yleiskäyttöisyys, käyttöönoton nopeus ja käytön helppous verrattuna täysin alusta asti kehitettyyn piiriin. Kehitysalustat soveltuvat kuitenkin huonosti tuotantoon ylimääräisten komponenttien, suuren kokonsa ja usein hintansa vuoksi. Lisäksi kaikkia kehitysalustoja ei välttämättä saa kopioida. Laadun ja tuottavuuden kannalta on parempi suunnitella käyttötarkoitukseen erikoistuneempi piiri mahdollisen kehitysalustalla luodun prototyypin pohjalta. [17]

3.1 Ohjausyksikön valinta ja PWM-signaali led-nauhan himmennyksessä

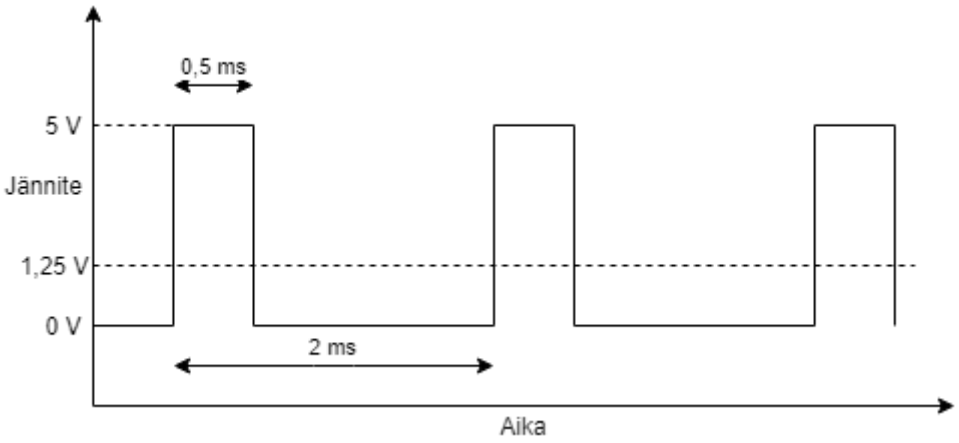
Tämän laitteen ohjaamiseen on valittu kehitysalusta Arduino Nano sen edullisuuden, helpon saatavuuden, aikaisemman kokemuksen alustan käytöstä sekä, esimerkiksi Arduino Unoon verrattuna, pienen koon vuoksi. Lisäksi Arduino Nano voidaan kiinnittää piikkirimojen avulla samalle piirilevylle muiden komponenttien kanssa.

Taulukko 2. *Arduino Nanon keskeisimmät ominaisuudet [18]*

Ominaisuus	**Arvo**
Käyttöjännite	5 V
Kellotaajuus	16 MHz
Flash-muisti	32 kt
SRAM	2 kt
EEPROM	1 kt
Yleiskäyttöiset digitaaliset I/O-pinnit	14 kpl (joista 6 PWM ulostuloja)
Analogiset sisäänmenot	8 kpl
Koko	18 x 45 mm

Kuten taulukosta 2 nähdään, Nanossa on kaikki IoT led -valon tarvitsemat toiminnallisuudet: yleiskäyttöisiä digitaalisia I/O-pinnejä, PWM-ulostuloja, analoginen sisäänmeno ja tarpeeksi muistia sekä laskentatehoa ohjelmaa varten.

Arduinon tehtävänä tässä laitteessa on led-nauhojen ohjaus transistorien avulla, laitteen fyysisen käyttöliittymän toiminnot, sarjaliikennekommunikointi WLAN-moduulin kanssa ja sarjaliikennekommunikointi tietokoneen kanssa debug-viestien lähettämistä varten. Mikäli debug-viestit on otettu ohjelmassa käyttöön, niiden avulla voidaan seurata sekä Arduinon, että ESP:n ohjelman kulkua, mikä helpottaa vikojen etsimistä ja korjaamista.



Kuva 5. *Arduinon tuottama PWM-signaali 1:3 pulssisuhteella (490 Hz) [19]*

Led-nauhaan kytkettyjä transistoreita ohjataan PWM-signaalin avulla. PWM-signaali on kanttiaalto, jonka pulssisuhde säädetään niin, että kunkin jakson keskiarvojännite vastaa haluttua arvoa. Tällöin signaalin keskiarvojännite vastaa myös haluttua arvoa. [19] Kuvassa 5 on esitetty Arduinon tuottama PWM-signaali 1:3 pulssisuhteella, jolloin jännitteen keskiarvo on 1,25 V. PWM-signaalin generointi tapahtuu mikrokontrollerin ajastimien avulla, se voi saada keskiarvojännitteekseen 0 V – 5 V, ja sen ohjaus tapahtuu ohjelmallisesti.

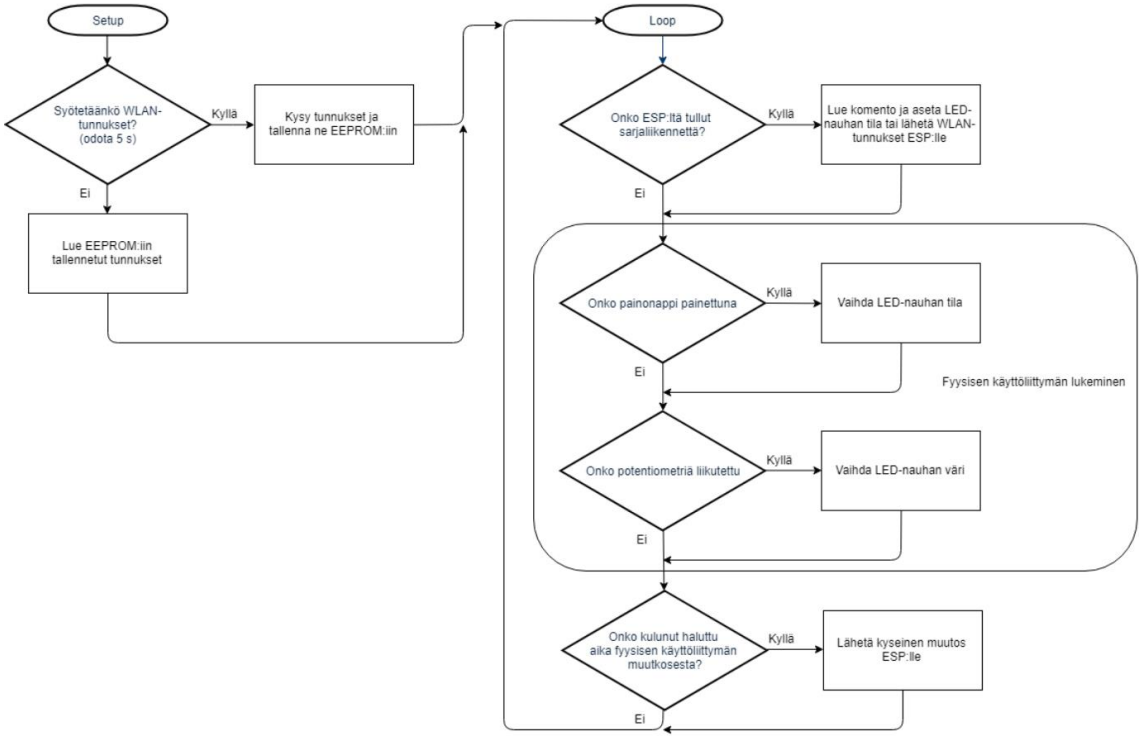
Kun PWM-signaali syötetään transistorin hilalle, hilan jännite vaihtelee signaalin maksimiarvon 5 V ja minimiarvon 0 V välillä pulssisuhteen mukaisesti. Transistorit on mitoitettu siten, että PWM-signaalin pulssin ollessa maksimiarvossaan transistorin kanava on johtavassa tilassa, ja pulssin ollessa minimiarvossaan transistorin kanava on kiinni. Tästä johtuen ledien kautta kulkeva virta on myös maksimiarvossaan tai sitä ei kulje ollenkaan, minkä takia ledit ovat vuorotellen kirkkaimmillaan tai pois päältä. Tällöin ledien keskimääräiseen kirkkauteen voidaan vaikuttaa niiden keskimääräistä virtaa ohjaamalla, eli ohjaamalla PWM-signaalin pulssisuhdetta. [13] Arduinon tuottaman PWM-signaalin taajuus on keskimäärin 490 Hz, mistä johtuen ledien vilkkumista ei huomaa paljaalla sil-

mällä, esimerkiksi tietokoneen näytöissä on usein 60 Hz päivitystaajuus, vaan ledit vaikuttavat palavan tasaisesti ja niiden kirkkauteen voidaan vaikuttaa PWM-signaalin pulssisuhteella [20, 21].

Arduinon tuottaman PWM-signaalin resoluutio on 8 bittiä, eli sen pulssisuhde voi saada $2^8 = 256$ eri arvoa (väliltä 0 V – 5 V). Näin ollen jokainen RGB-ledien kolmesta eri väristä voi saada 256 eri kirkkaustasoa, eli teoriassa näin pystytään tuottamaan $256^3 = 16\,777\,216$ väriä. Käytännössä kuitenkin monet näistä väreistä ovat hyvin lähellä toisiaan, eikä niitä kaikkia voida luoda tarkasti, koska ledien kirkkaus, kynnysjännite ja niiden tuottaman valon aallonpituus vaihtelevat merkittävästi yksittäisten ledien välillä. Lisäksi ledien ominaisuudet muuttuvat ajan ja lämpötilan myötä. Tarkkaa värintuottoa varten laitteessa tulisi olla takaisinkytkentäjärjestelmä, mutta koska eri värejä voidaan tuottaa tarpeeksi tämän laitteen pääasiallista käyttökohdetta eli tunnelmavalaistusta varten ilman takaisinkytkentää, ei tällainen järjestelmä ole tarpeen. [22]

3.2 Arduinon ohjelmointi

Arduino-ohjelmointikieli perustuu mikrokontrollerien ohjelmointiin tarkoitettuun Wiring-kieleen, joka puolestaan perustuu C- ja C++-kieliin ja joka käännetään C++-kääntäjällä. Tästä johtuen Arduino-koodin joukossa voidaan käyttää myös C:n ja C++:n funktioita ja tietotyyppejä. [16, 23] Arduino-kielen käyttö on helpompaa ja selkeämpää kuin pelkän C-kielen, jonka käyttäminen AtMega328p:n ohjaamiseen tapahtuu asettamalla ja lukemalla mikrokontrollerin ohjaus- ja statusrekistereitä [6]. Arduino-kielen käyttäminen vaatii kuitenkin enemmän mikrokontrollerin laskentatehoa ja muistia kuin C, sillä se on korkeamman tason ohjelmointikieli, joten mikäli ohjelman on säästettävä resurssien käyttöä, on C-kieli parempi vaihtoehto. Tämän laitteen ohjelmisto on kuitenkin yksinkertainen, ja käyttää vain osan mikrokontrollerin resursseista, joten se on kirjoitettu Arduino-kielellä, kuitenkin käyttäen vain C:n tietotyyppejä. Arduino-ohjelma, jota kutsutaan nimellä sketch, koostuu kahdesta osasta, joiden nimet ovat setup ja loop. Setup-osassa oleva ohjelmakoodi suoritetaan kerran mikrokontrollerin käynnistyessä, eli siinä suoritetaan tarvittavat alustustoimenpiteet, ja loop-osa on silmukkarakenne, jossa suoritetaan ohjelman varsinainen toiminta.



Kuva 6. *Arduinon ohjelman toiminta vuokaaviona*

Tässä laitteessa Arduino kysyy ensiksi WLAN-tunnukset, pollaa fyysistä käyttöliittymää, lukee ja kirjoittaa kahta sarjaliikenneväylää sekä syöttää kolmelle transistorille PWM-signaalia. Nämä kaikki toiminnot ovat yksinkertaisia ohjelmoida Arduino-kielellä, ja ne on esitetty kuvassa 6 vuokaaviona.

```
//Vakioden määrittely ja ohjelmallisen sarjaliikenneolion luominen oh
//jelman alussa:
#define PUN_PIN 3
#define VIH_PIN 5
#define SIN_PIN 11
#define ONOFF_PIN 2
#define POT_PIN A0
#define POT_TOL 10
#define JULKAISUVALI 5000 //(ms)
//WLAN-salasanan ja SSID:n maksimipituus
#define TUNNUS_MAX_KOKO 32 + 1
#define SSID_OSOITE 0                           //EEPROM-osoite
#define SALASANA_OSOITE TUNNUS_MAX_KOKO

//Jos DEBUG on tosi, lähetetään debug-viestit tietokoneelle DEBUGP
//RINTLN-makrolla
#define DEBUG true
#define DEBUGPRINTLN(x)    if (DEBUG) { Serial.print("Arduino>>> ");
Serial.println(x); }

char ledTila[] = "OFF"; //Led-nauhan tila merkkitaulukkona
uint8_t ledRGB[] = {150, 150, 150}; //Led-nauhan väri taulukkona 8
                                    //bittisiä lukuja
char ssid[TUNNUS_MAX_KOKO];
char wlanSalasana[TUNNUS_MAX_KOKO];
--- Muiden globaalien muuttujien alustus ---

SoftwareSerial soft(6, 7); // RX, TX

//setup-osio:
void setup() {
  // I/O-pinnien alustukset
  pinMode(PUN_PIN, OUTPUT);
  pinMode(VIH_PIN, OUTPUT);
  pinMode(SIN_PIN, OUTPUT);

  pinMode(ONOFF_PIN, INPUT);
  pinMode(POT_PIN, INPUT);

  soft.begin(9600);
  Serial.begin(9600);

  //aseta debug serial
  if ( DEBUG ){
    debugserial = &Serial;
  } else {
    debugserial = NULL;
  }
  DEBUGPRINTLN("Alustettu");}

  --- WLAN-tunnuksien syöttäminen ---

  potTila = analogRead(POT_PIN);
}
```

Ohjelma 1. *Arduino-ohjelman vakioiden määrittelyt ja setup-osio*

Ohjelman alussa, joka on esitetty ohjelmassa 1, määritellään tarvittavien I/O-pinnien numerot vakioksi, alustetaan tarvittavat globaalit muuttujat ja luodaan SoftwareSerial-olio ESP8266:n kanssa kommunikointia varten. Setup-osassa alustetaan I/O-pinnit tarpeen mukaan joko sisäänmenoiksi tai ulostuloiksi, jonka jälkeen avataan sarjaliikenneyhteys sekä tietokoneeseen, että ESP:hen. Molemmissa tapauksissa käytetään baudinopeutta 9600. Tämän jälkeen Arduino kysyy WLAN-tunnuksia käyttäjältä. Mikäli käyttäjä syöttää 5 s kuluessa ohjelman alusta merkin ’k’ Arduinon sarjaliikenneporttiin esimerkiksi Arduino IDE:n sarjaliikennemonitorista, odottaa Arduino kunnes sekä WLAN:in verkkotunnus että salasana on vastaanotettu ja tallentaa ne sitten EEPROM-muistiin (engl. Electronically Erasable Programmable Read-Only Memory, haihtumaton puolijohdemuisti). Mikäli ohjelman alusta kuluu 5 s ja merkkiä ’k’ ei ole vastaanotettu, luetaan viimeksi syötetyt tunnukset EEPROM:ista globaaleihin muuttujiin ssid ja wlanSalasana, jotta ne voidaan lähettää ESP:lle tarvittaessa. Aikaraja tunnusten syöttämisen aloittamiseksi on asetettu, jotta laite voi aloittaa toimintansa vaikka se ei olisi kytkettynä tietokoneeseen. Lopuksi luetaan potentiometrin alkutila.

```
--- ESP:ltä saapuneen komennon lukeminen loop-osiossa: ---
char komento[TUNNUS_MAX_KOKO + 1];

if ( lueKomento(&soft, komentomerkit, komento, debugserial) ) {
  //ESP:ltä on saapunut komento
  char *pKomento = komento;
  if (pKomento [0] == 'L' ) {
    //Komento koskee led-nauhan tilaa
    ++pKomento; //Siirry parametrimerkkijonoon
    //Vaihda led-nauhan tila saadun parametrin perusteella
    if ( strcmp(pKomento, "OFF") == 0 ) {
      sprintf(ledTila, "%s", pKomento);
    } else if ( strcmp(pKomento, "ON") == 0 ) {
      sprintf(ledTila, "%s", pKomento);
    }
    DEBUGPRINTLN(ledTila);
    paivitaLED(ledTila, ledRGB);
  } else if (pKomento [0] == 'C' ) {
    //Komento koskee led-nauhan väriä
    ++pKomento;
    char tmp[3];
    for ( int n = 0 ; n < 3 ; ++n ) {
      //Lue komennon kaksi ensimmäistä merkkiä
      sprintf(tmp, "%.2s", pKomento);
      //Tulkitse merkit hexadesimaaleista kymmenluvuiksi
      //ja aseta ne väritaulukkoon ledRGB
      ledRGB[n] = (int)strtol(tmp, NULL, 16);
      pKomento += 2; //Siirry kaksi merkkiä eteenpäin
    }

    DEBUGPRINTLN("color:");
    DEBUGPRINTLN(ledRGB[0]);
    DEBUGPRINTLN(ledRGB[1]);
    DEBUGPRINTLN(ledRGB[2]);

    paivitaLED(ledTila, ledRGB);

  } else if ( pKomento[0] == 'G' ){
      //ESP pyytää wlan-tunnuksia
      lahetaKomento(&soft, 'S', ssid);
      delay(5);
      lahetaKomento(&soft, 'P', wlanSalasana);
  } else if ( pKomento[0] == 'G' ){
      //ESP pyytää wlan-tunnuksia
      lahetaKomento(&soft, 'S', ssid);
      delay(5);
      lahetaKomento(&soft, 'P', wlanSalasana);
    }
}
```

Ohjelma 2. *ESP:ltä saapuneen komennon lukeminen*

Loop-osio, eli varsinainen toiminnallisuus alkaa mahdollisen ESP:ltä tulevan komennon lukemisella ohjelmallisesta sarjaliikenneportista funktiolla lueKomento ohjelmassa 2. Funktio palauttaa osoittimen merkkitaulukkoon, jonka ensimmäinen merkki vastaa komentoa, joka halutaan suorittaa ja loput merkit parametriä tälle komennolle. Ensimmäisen merkin ollessa ’L’, vaihdetaan led-nauhan tilaa päälle/pois (parametrinä merkkijono

”ON” tai ”OFF”), sen ollessa ’C’, vaihdetaan led-nauhan väriä (parametrinä merkkijono, jossa on kaksi heksadesimaalimerkkiä kutakin väriä kohden) ja sen ollessa ’G’, lähetetään WLAN-tunnukset ESP:lle. Led-nauhan tila luetaan yksinkertaisesti tarkastamalla, kumpi mahdollisista parametreistä on saatu, ja sen väri luetaan lukemalla parametrimerkkijono kaksi merkkiä kerrallaan, jotka sitten muutetaan heksadesimaalijärjestelmästä kymmenlukujärjestelmään. Molemmissa tapauksissa saatu parametri tallennetaan globaaliin muuttujaan, ja kutsutaan paivitaLED-funktiota, joka toteuttaa muutokset.

```
--- Painonapin tilan lukeminen loop-osiossa: ---
if ( digitalRead(ONOFF_PIN) == HIGH ) {
  //Painonappi on painettuna, vaihda led-nauhan tila
  if ( strcmp(ledTila, "OFF") == 0 ) {
    sprintf(ledTila, "%s", "ON");
  } else if ( strcmp(ledTila, "ON") == 0 ) {
    sprintf(ledTila, "%s", "OFF");
  }
  viimOnoffMuutos = millis(); //tallennetaan muutoksen ajankohta
  onoffMuutos = true;
  delay(300);
  DEBUGPRINTLN(ledTila);
  paivitaLED(ledTila, ledRGB);
}
```

Ohjelma 3. *Painonapin tilan lukeminen*

Seuraavaksi painonapin tila luetaan ohjelmassa 3. Painonapin ollessa painettuna siihen kytketyn I/O-pinnin jännite on 5 V. Funktio digitalRead palauttaa vakion HIGH, jos pinnin jännite on yli 3.0 V, muutoin se palauttaa vakion LOW. Kun painonapin I/O-pinnin jännite on 5 V, vaihdetaan led-nauhan tila päälle, jos se oli alun perin pois päältä, ja pois päältä, jos se oli alun perin päällä. Tämän jälkeen odotetaan 300 ms tekemättä mitään, jotta I/O-pinnin jännite ehtii tasaantua takaisin nollaan volttiin, eikä yhdestä painalluksesta aiheudu useita led-nauhan tilan muutoksia. Lopuksi kutsutaan paivitaLED-funktiota.

```
int potTila = 0;
uint8_t variNro = 0;
uint8_t variMaara = 7;    //Värien maksimimäärä
uint8_t variTaulukko[][4] = { {0, 255, 0}, {255, 0, 0}, {0, 0, 255},
{255, 255, 0}, {0, 255, 255}, {255, 0, 255}, {255, 255, 255} };

--- Potentiometrin tilan lukeminen loop-osiossa: ---
int nykPotTila;
nykPotTila = analogRead(POT_PIN); //Lue potentiometrin tämänhetkinen
//tila
if ( (potTila - POT_TOL) > nykPotTila || nykPotTila > (potTila +
POT_TOL) ){
    //Potentiometriä on liikutettu
    potTila = nykPotTila;

    //Laske missä potTaulukon värissä ollaan
    uint8_t nykVariNro = 0;
    if (potTila - 4 < 0) potTila -= potTila -4;
    nykVariNro = (potTila - 4) / ( 1023 / variMaara);
    //nykVariNro välillä 0 - (variMaara - 1 ), siksi -4

    //kerroin värin kirkkauden skaalaamiseen potentiometrin asennon
    //mukaan
    float kerroin = ( (potTila - 4) % (1023 / variMaara)  ) /
                    (1023.0 / variMaara);

    for ( uint8_t i = 0; i <= 2; ++i) {
      //Aseta led-nauhan väri
      ledRGB[i] = variTaulukko[nykVariNro][i];
      //Skaalaa kirkkaus
      ledRGB[i] = ledRGB[i] / 2.0 + (ledRGB[i] / 2.0) * kerroin;
    }

    paivitaLED(ledTila, ledRGB);
    DEBUGPRINTLN("RGB: ");
    DEBUGPRINTLN(ledRGB[0]);
    DEBUGPRINTLN(ledRGB[1]);
    DEBUGPRINTLN(ledRGB[2]);

    //Mikäli väri vaihtuu, voidaan muutos lähettää ESP:lle
    if ( nykVariNro != variNro ){
      variNro = nykVariNro;
      potMuutos = true;

    }

    viimPotMuutos = millis();
  }
```

Ohjelma 4. *Potentiometrin tilan lukeminen*

Ohjelmassa 4 on esitetty potentiometrin tilan lukeminen loop-osiossa. Potentiometrin keskimmäinen liitin on kytketty Arduinon analogiseen sisäänmenoon A0. Tämän pinnin jännitteen muutoksesta huomataan siis, onko potentiometriä säädetty eli vaihdetaanko led-nauhan väriä. Jännitteen alkutila on mitattu setup-osiossa muuttujaan potTila ja sen arvot ovat väliltä 0 – 1023. Jokaisella loop-silmukan kierroksella jännite mitataan uudes-

taan ja tallennetaan väliaikaiseen muuttujaan, jota verrataan sen edelliseen potTila-muuttujaan tallennettuun arvoon. Alussa määritelty vakio POT_TOL on toleranssi, jonka sisällä jännitteen muutosta ei huomioida. Jos mitattu arvo on suurempi kuin potTila + POT_TOL tai jos sen on pienempi kuin potTila - POT_TOL, tulkitaan, että potentiometriä on liikutettu, ja mitattu arvo tallennetaan potTila-muuttujaan. Tämän jälkeen potentiometriltä luettu arvo tulkitaan väriksi. Ohjelman alussa asetettu muuttuja variTaulukko sisältää n (käytetty arvoa 7) väriä kaksiulotteisena taulukkona, ja potentiometriä kääntämällä voidaan valita mikä tahansa näistä väreistä. Tämä tapahtuu siirtämällä potentiometriltä luettu arvo välille 0 – (n-1) (variNro) ja kopioimalla sitten variTaulukon väri vastaavalla indeksillä ledRGB-taulukkoon. Myös värien kirkkauteen voidaan vaikuttaa potentiometrin asennolla. Potentiometriltä luetun arvon kasvaessa värin kirkkaus kasvaa, kunnes väri vaihtuu. Tämä saadaan aikaan skaalaamalla väriarvojen puolikkaat kertoimella, joka on laskettu siten, että sen arvo kasvaa lineaarisesti arvosta 0 arvoon 1 variNroiden välillä, ja lisäämällä sitten näihin arvoihin väriarvojen puolikkaat. Värien suhteellinen minimikirkkaus on siis 0,5 (kerroin on 0) ja maksimikirkkaus 1,0 (kerroin on 1). Värien muutokset julkaistaan IoT-alustalle (lähetetään ESP:lle) vain, kun väri vaihtuu, ei kirkkauden vaihtuessa.

```
--- Komentojen lähetys ESP:lle loop-osiossa: ---

if ( onoffMuutos == true ) {
  if ( millis() - viimOnoffMuutos > JULKAISUVALI) {
    //Lähetä komento ESP:lle
    DEBUGPRINTLN("ONOFF Julkaisu");
    lahetaKomento(&soft, 'L', ledTila);
    onoffMuutos = false;
  }
}

if ( potMuutos == true) {
  if ( millis() - viimPotMuutos > JULKAISUVALI) {
    //Lähetä komento ESP:lle
    char parametri[7];
    //Muuta tallennetut väriarvo heksadesimaalimerkeiksi (2 merkkiä/arvo)
    sprintf(parametri, "%.2x%.2x%.2x", ledRGB[0], ledRGB[1],
            ledRGB[2]);
    DEBUGPRINTLN("Vari julkaisu");
    lahetaKomento(&soft, 'C', parametri);
    potMuutos = false;
   }
}
```

Ohjelma 5. *Komentojen lähetys ESP:lle*

Fyysisellä käyttöliittymällä tehdyt muutokset led-nauhan tilassa ja värissä lähetetään ESP:lle ja siten IoT-alustalle ohjelmassa 5 vasta, kun niiden edellisestä muutoksesta on kulunut määritelty aika, sillä niiden päivittyminen IoT-alustalle heti, kun ne on tehty, ei ole tarpeellista ja niiden jatkuva lähettäminen olisi turhaa liikennettä. Lisäksi IoT-alusta

lähettää sille saapuneen komennon takaisin viiveellä, joten muutosten jatkuva lähettäminen aiheuttaisi tilanteen, jossa fyysisellä käyttöliittymällä tehdyt nopeat muutokset välittyisivät led-nauhalle ensin suoraan Arduinolta, ja sitten viiveellä IoT-alustan lähettäminä. Tämä ongelma voitaisiin ratkaista vaihtamalla led-nauhan tilaa vain IoT-alustalta saaduilla käskyillä, mutta silloin menetettäisiin laitteen kyky toimia ilman verkkoyhteyttä. Viive toteutetaan kahdella muuttujalla led-nauhan tilaa ja väriä kohden. Molemmilla ominaisuuksilla on muuttuja, joka kertoo, onko muutoksia tehty, ja muuttuja, joka kertoo koska edellinen muutos on tehty tallentamalla millis-funktion antaman ajan Arduinon käynnistymisestä millisekunteina. Kun tarkastetaan, julkaistaanko muutosta, tarkastetaan että muutos on tehty, mikä saadaan suoraan muuttujasta potMuutos tai onoffMuutos (arvo on tosi, jos muutos on tehty), ja että muutoksesta on kulunut haluttu aika, joka saadaan vähentämällä viimeisimmän muutoksen tekohetkellä tallennettu aika vertailuhetken ajasta. Jos tämä erotus on suurempi kuin määritelty julkaisuväli, lähetetään komento eteenpäin ESP:lle, ja asetetaan potMuutos tai onoffMuutos epätodeksi, jotta samaa muutosta ei lähetetä useaan kertaan.

```
void paivitaLED(const char* ledTila, const uint8_t* rgb) {
  DEBUGPRINTLN("PaivitaLED");
  if ( strcmp(ledTila, "ON") == 0 ) {
    digitalWrite(LED_BUILTIN, HIGH); //Jos led-nauha on päällä, se
    //nähdään myös Arduinon sisäänrakennetusta ledistä
    analogWrite(PUN_PIN, rgb[0]);
    analogWrite(VIH_PIN, rgb[1]);
    analogWrite(SIN_PIN, rgb[2]);
  } else if ( strcmp(ledTila, "OFF") == 0 ) {
    digitalWrite(LED_BUILTIN, LOW);
    analogWrite(PUN_PIN, 0);
    analogWrite(VIH_PIN, 0);
    analogWrite(SIN_PIN, 0);
  }
}
```

Ohjelma 6. *Funktio led-nauhan tilan ja värin päivittämiseen*

Led-nauhan tilan ja värin vaihtuminen suoritetaan paivitaLED-funktiolla, joka on esitetty ohjelmassa 6. Funktiolle annetaan parametreinä led-nauhan tila ja väri, ja se asettaa määriteltyihin I/O-pinneihin PWM-signaalin näiden perusteella analogWrite-funktiolla. Koska led-nauhan väri tallennetaan kolmen 8 bittisen luvun taulukkona, voidaan tämän taulukon arvoja käyttää suoraan analogWrite-funktion parametrinä PWM-signaalin pulssisuhteelle.

3.3 Sarjaliikennekommunikaatio Arduinon ja WLAN-moduulin välillä

Sekä Arduino, että ESP käyttävät samoja tähän käyttötarkoitukseen suunniteltuja funktioita komentojen lähettämiseen ja vastaanottamiseen sarjaliikenteen avulla. Molemmissa on sisäänrakennettu Universal Asynchronous Receiver Transmitter (UART, universaali

asynkroninen lähetin-vastaanotin) -oheislaitelohko, jolla tämä sarjaliikennekommunikaatio toteutetaan.

Käskyt lähetetään ASCII-merkkeinä, ja ne ovat muotoa <alkumerkki><komentomerkki><parametrimerkkijono><loppumerkki>. Alku- ja loppumerkeistä tunnistetaan komennon alku ja loppu, jotta komento erotetaan muusta sarjaliikenteestä, ja ne määritellään vakioksi CMD_START ja CMD_STOP. Komentomerkki kertoo, mitä ominaisuutta komento koskee, ja parametrimerkkijono on tälle ominaisuudelle asetettava arvo.

```
void lahetaKomento(Stream *serial, char komentomerkki,
                   char parametri[]) {
  //Funktio, joka lähettää komennon sarjaliikenneyhteydellä
  char komento[TUNNUS_MAX_KOKO + 3];
  sprintf(komento, "%c%c%s%c", CMD_START, komentomerkki, parametri,
  CMD_STOP)

  serial->print(kom);
}
```

Ohjelma 7. *Komennon lähetysfunktio*

Ohjelmassa 7 esitetylle komennon lähetysfunktiolle annetaan parametreinä osoitin sarjaliikenneolioon, komentomerkki ja parametrimerkkijono merkkitaulukkona. Funktio muodostaa saamistaan parametreistä ja alku- ja loppumerkkivakioista lähetettävän komennon ja lähettää sen sarjaliikenneolion avulla kutsumalla tämän metodia print. Taulukossa 3 on esitetty kaikki tässä laitteessa käytetyt komennot ja niiden komentomerkit.

Taulukko 3. *Sarjaliikennekommunikaatiossa käytetyt komennot*

Kuvaus	**Parametri**	**Komentomerkki**
Led-nauha päälle/pois päältä	”ON” / ”OFF”	L
Led-nauhan värin asetus	<HEX-värikoodi>	C
WLAN-tunnusten kyselykomento	Ei käytössä	G
WLAN-verkkotunnus	<verkkotunnus>	S
WLAN-salasana	<salasana>	P

```
bool lueKomento(Stream *serial, char komentomerkit[], char *mjono,
Stream *debugSerial /*= NULL*/ ) {
  //Funktio, joka lukee komennon sarjaliikenneväylältä
  if ( serial->available() ) {
    while (serial->available()) {
      char c1 = (char)serial->read();
      if (  c1 == CMD_START ) {
        //Ensimmäinen merkki on alkumerkki
        delay(10);
        char c2 = (char)serial->read();
        //Käy komentomerkit läpi
        for ( int i = 0 ; i < strlen(komentomerkit) ; ++i ) {
          if ( c2 == komentomerkit[i] ) {
            //Seuraava merkki on komentomerkki
            mjono[0] = komentomerkit[i];
            delay(10); //Odota loppukomennon saapumista puskuriin
            //Lue merkkejä kunnes komennon maksimipituus ylitetään,
            //loppumerkki löytyy tai sarjaliikennepuskuri on tyhjä
            for ( int n = 1 ; n < TUNNUS_MAX_KOKO + 2 ; ++n) {
              char c3 = (char)serial->read();
              if (c3 == CMD_STOP) {
                //Loppumerkki löytyi
                mjono[n] = '\0'; //Tallenna loppumerkin si
                                 //jasta \0
                return true;
              } else if ( n == TUNNUS_MAX_KOKO ){
                //Parametrin maksimipituus on ylitetty
                //tulosta luetut merkit
                debugSerial->print(c1);
                debugSerial->print(mjono);
                debugSerial->print(c3);
              } else {
                //c3 kuuluu parametrimerkkijonoon
                mjono[n] = c3;
              }
            }
            return false;    //parametrin maksimipituus ylitetty
          }
        }
        //Komentomerkkiä ei löytynyt, tulosta luetut merkit
        if (debugSerial != NULL)
          debugSerial->print(c1);
          debugSerial->print(c2);
        return false;

      } else if ( debugSerial != NULL) {
        //Alkumerkkiä ei löytynyt, tulosta merkki
        debugSerial->print(c1);
      }
    }
  }
  //Sarjaliikennebufferi on tyhjä, komentoa ei löytynyt
  return false;
}

```

Ohjelma 8. *Komennon vastaanottofunktio*

Komennon vastaanottofunktio, joka on esitetty ohjelmassa 8, ottaa parametreinään osoittimen sarjaliikenneolioon, josta komennot luetaan (serial), merkkitaulukon, joka sisältää käytetyt komentomerkit (komentomerkit), osoittimen merkkitaulukkoon, johon komento tallennetaan (mjono) sekä osoittimen sarjaliikenneolioon, jonka avulla lähetetään komentoon liittymättömät merkit eteenpäin (debugSerial). Viimeiselle parametrille on määritelty oletusarvo NULL, sillä sitä ei ole välttämätöntä käyttää. Funktio asettaa mjono-osoittimen osoittaman merkkitaulukon siten, että sen ensimmäinen merkki on luettu komentomerkki ja loput merkit muodostavat parametrimerkkijonon. Mikäli komento on luettu onnistuneesti funktio palauttaa arvon tosi, ja jos sarjaliikennepuskuri on tyhjä tai ei sisällä komentoa, se palauttaa arvon epätosi.

Funktio toimii vertaamalla sarjaliikennepuskuriin saapuneita merkkejä komennon alkumerkkiin. Jos saapunut merkki ei vastaa alkumerkkiä, se kirjoitetaan debug-sarjaliikenneporttiin, jos sellainen on määritelty ja luetaan seuraava merkki. Alkumerkin löytyessä luetaan sitä seuraava merkki, jota puolestaan verrataan komentomerkkitaulukon merkkeihin. Jos tämä toinen merkki on komentomerkkitaulukossa, asetetaan se mjono-merkkitaulukon ensimmäiseksi merkiksi ja odotetaan 10 ms, jotta loput komennon merkeistä saapuvat sarjaliikennepuskuriin. Näitä parametrimerkkijonon merkkejä luetaan ja tallennetaan mjono-merkkitaulukkoon, kunnes luettu merkki vastaa loppumerkkiä tai on luettu palautettavan merkkijonon maksimipituuden verran merkkejä. Loppumerkin sijasta palautettavaan merkkitaulukkoon tallennetaan taulukon loppua kuvaava merkki ’\0’, ja palautetaan tosi. Jos loppumerkkiä ei löydy ennen kuin komennon maksimipituus ylitetään, palautetaan epätosi. Epätosi palautetaan myös, jos päästään funktion loppuun, eli sarjaliikennepuskuri on funktion alussa tyhjä tai se on luettu tyhjäksi.

4. WLAN-MODUULIN TOIMINTA

Laitteeseen on valittu langaton internetyhteys langallisen sijasta, sillä sen halutaan edustavan tyypillistä kodin IoT-sovellusta, joissa langaton yhteys on hyvin yleinen kayttäjäystävällisyytensä vuoksi sen mahdollistaessa laitteen vapaamman sijoittelun. Luotettavuuden, turvallisuuden ja yhteyden nopeuden kannalta langallinen verkkoyhteys olisi parempi vaihtoehto, mutta langattomallakin yhteydellä päästään näissä asioissa tarpeeksi hyvälle tasolle, jotta langaton yhteys on pätevä vaihtoehto tämän kaltaiseen sovellukseen, jossa nämä seikat eivät ole kriittisiä.

4.1 WLAN-moduulin tekniset tiedot ja sen ohjelmointi

Tässä laitteessa käytössä oleva Ai-Thinkerin ESP8266 ESP-01S on edullinen Espressif Systemsin ESP8266EX-mikrokontrolleriin perustuva WLAN-moduuli, jota voidaan joko ohjelmoida suoraan tai sitä voidaan ohjata toisen mikrokontrollerin avulla. Sen WLAN-taajuus on 2,4 GHz ja se koostuu muun muassa ESP8266EX:stä, piirilevyantennista ja 8-pinnisestä liittimestä. [7, 24]

ESP8266-moduulista on saatavilla useita eri malleja ja kehitysalustoja, jotka hyödyntävät ESP8266EX:n ominaisuuksia vaihtelevasti. ESP-01S-malli on ensisijaisesti tarkoitettu käytettäväksi toisen mikrokontrollerin oheislaitteena, ja siinä onkin vain 8 pinniä liityntöjä varten, jonka takia osa ESP8266EX-kontrollerin ominaisuuksista ei ole käytössä. Toinen yleisesti käytössä oleva itsenäiseen toimintaan tarkoitettu malli on ESP-12 kehitysalusta, joka tarjoaa pääsyn useampiin ESP8266EX:n ominaisuuksiin kuin ESP-01S, mutta sen hinta on korkeampi. Koska ESP-moduulia käytetään tässä sovelluksessa yhdessä Arduinon kanssa, on ESP-01S halvemman hintansa perusteella sopivampi valinta. [7]

Taulukko 4. *ESP8266 ESP-01S pinnien kuvaukset ja kytkennät [7, 24]*

Pinni	Kytkentä	Kuvaus
RX	Arduino D7 (TX)	UART sarjaliikenne, vastaanotto
TX	Arduino D6 (RX)	UART sarjaliikenne, lähetys
VCC	3,3V	Käyttöjännite 2,5V – 3,6V
GND	Arduino GND	Maataso
GPIO0	GND tai (-)	Kytketään maahan ohjelmointia varten ja käyttöjännitteeseen itsenäistä toimintaa varten
GPIO2	-	Yleiskäyttöinen I/O-pinni
CH_PD	-	Kytkee moduulin päälle (VCC) tai pois päältä (GND)
RST	-	Ulkoinen reset-signaali, aktiivinen alasvietynä

Taulukon 4 kytkentä-sarakkeessa on esitetty pinnien kytkentä tässä laitteessa. GPIO0-, CH_PD- ja RST-pinneillä on ESP-01S-mallissa sisäiset ylösvetovastukset, joten ne jätetään kytkemättä, mikä on esitetty taulukossa viivalla [7]. GPIO2-pinni jätetään kytkemättä, koska sitä ei tarvita.

ESP-01:en toimiessa toisen mikrokontrollerin alaisuudessa, sitä voidaan ohjata AT-komennoilla, jotka ovat ESP8266EX:n sarjaliikenneporttiin lähetettäviä merkkijonoja. Ne voidaan jakaa neljään eri luokkaan: testi-, kysely-, asetus- ja suorituskomennot, ja ne noudattavat yleisesti formaattia AT+<KOMENTO><X>. Kyselykomennot palauttavat halutun parametrin, esimerkiksi WLAN-yhteyden statuksen. Asetuskomennoilla asetetaan parametrien arvoja, esimerkiksi UART:in baudinopeus, ja näiden parametrien mahdolliset arvot saadaan selville testikomennoilla. Suorituskomennot suorittavat jonkin toiminnon. Samasta komennosta voi olla yksi tai useampia näihin luokkiin kuuluvaa versiota. [25]

Tässä laitteessa ESP:n tehtävänä on luoda WLAN-yhteys, kommunikoida IoT-alustan kanssa ja syöttää tätä kautta saadut komennot Arduinolle, joka sitten ohjaa led-nauhaa. ESP myös vastaanottaa Arduinon lähettämiä komentoja sen fyysiseltä käyttöliittymältä, ja välittää nämä sitten IoT-alustalle. ESP:tä ei siis ohjata AT-komennoilla, vaan se ohjelmoidaan Arduino IDE:n avulla suorittamaan nämä toiminnot itsenäisesti käyttämällä kolmannen osapuolen ESP8266 Arduino corea, joka mahdollistaa ESP:n ohjelmoinnin Arduino-ohjelmointikielellä [26].

ESP:n ohjelmointi tapahtuu sarjaliikennepinnien RX ja TX kautta. Helpoin tapa ohjelmoida ESP on käyttämällä Arduino IDE:ä ja Arduinoa. ESP kytketään taulukon 3 mukaisilla kytkennöillä, eli ESP:n TX kytketään Arduinon TX:än ja ESP:n RX Arduinon RX:än, jotta Arduinon USB-liityntään saapuva sarjaliikenne saadaan välitettyä ESP:lle. GPIO0-pinni on myös kytkettävä maahan moduulin käynnistyessä, jotta ESP kytkeytyy ohjelmointimoodiin. Lisäksi Arduinon reset-pinni kytketään maahan, jotta Arduinon mikroprosessori ei kytkeydy päälle ja Arduinon USB-UART-muunnin vain välittää sarjaliikenteen ESP:lle. Kun ESP ohjelmoidaan näin, AT-käyttöliittymä ylikirjoitetaan, eli sitä ei voida käyttää.

4.2 WLAN-yhteyden muodostaminen

WLAN muodostetaan tukiaseman avulla, jonka jälkeen siihen voidaan liittää haluttuja laitteita. Tukiasema, ja siihen liitetyt laitteet, voidaan edelleen liittää langallisella yhteydellä internettiin reitittimen avulla. ESP-moduulia voidaan käyttää joko tukiasemana soft access point -moodissa (soft AP) tai se voidaan liittää tukiasemaan station-moodissa (STA). Soft AP -moodissa ESP:n luomaan lähiverkkoon voidaan liittää muita laitteita, esimerkiksi älypuhelimia, mutta ESP ei tarjoa rajapintaa langalliselle internetyhteydelle, joten tämän lähiverkon kautta ei päästä suoraan internettiin. Tätä moodia voidaan kuitenkin käyttää sovelluksissa, joissa ESP:tä ohjataan langattoman lähiverkon kantaman sisäpuolelta tai joissa lähiverkkotunnukset annetaan ensin jollakin laitteella ESP:lle, jonka

jälkeen ESP yhdistetään tukiasemaan STA-moodissa. Soft AP - ja STA-moodeja voidaan myös käyttää samanaikaisesti. [27] Tässä laitteessa ESP toimii STA-moodissa, eli se yhdistetään olemassa olevan tukiaseman kautta internettiin.

WLAN-yhteyden muodostamiseen käytetään ESP8266 Arduino coreen kuuluvaa ESP8266WiFi-kirjastoa, jonka suunnittelu ja nimeämiskäytäntö ovat samanlaiset kuin Arduinon omassa WiFi-kirjastossa. [27]

```
--- WLAN-yhteyden luominen setup-osiossa: ---
DEBUGPRINTLN("Yhdistetaan:");
DEBUGPRINTLN(VERKKO_NIMI);

WiFi.mode(WIFI_STA); //Aseta WLAN-moodi STA

WiFi.begin(); //ALoita WLAN-yhteys flash-muistiin tallennetuilla tun
              //nuksilla
//WiFi.status() palauttaa WL.CONNECTED, mikäli verkkoyhteys on luotu
while ( WiFi.status() != WL_CONNECTED ){
  uint8_t i = 0;
  while (WiFi.status() != WL_CONNECTED && i < 40) {
    //Yritä yhdistää muistiin tallennetuilla tunnuksilla 20s
    delay(500);
    DEBUGPRINTLN(".");
    ++i;
  }
  //Vanhoilla tunnuksilla yhdistäminen ei onnistunut, kysy nykyiset
  //tunnukset Arduinolta
  kysyTunnukset(ssid,salasana);

  DEBUGPRINTLN("Uudet tunnukset saatu:");
  DEBUGPRINTLN(&ssid[1]);
  DEBUGPRINTLN(&salasana[1]);
  //Yhdistä käyttäen uusia tunnuksia (ensimmäinen merkki on komento
  //merkki)
  if ( WiFi.status() != WL_CONNECTED)
    WiFi.begin(&ssid[1],&salasana[1]);
}
DEBUGPRINTLN();
DEBUGPRINTLN("WLAN-yhteys luotu, IP osoite:");
DEBUGPRINTLN(WiFi.localIP());
```

Ohjelma 9. *ESP8266-moduulin yhdistäminen langattomaan lähiverkkoon*

WLAN-yhteys muodostetaan ESP:n ohjelman setup-osiossa, esitetty ohjelmassa 9, asettamalla ESP ensiksi STA-moodiin. Sitten lähiverkkoyhteys aloitetaan WiFi-luokan begin-metodilla, joka ottaa parametreinään yhdistettävän WLAN:in nimen ja salasanan, tai ei mitään, jolloin yhteys aloitetaan aikaisemman yhdistyksen yhteydessä automaattisesti ESP:n flash-muistiin tallentamilla tunnuksilla. Tämän jälkeen odotetaan 20 s yhteyden luomista ja mikäli se ei onnistu, kysytään nykyiset tunnukset Arduinolta kysyTunnukset-funktiolla. Tämä funktio lähettää Arduinolle WLAN-tunnusten kyselykomennon laheta-Komento-funktiolla, ja sitten kutsuu lueKomento-funktiota kahdesti, lukien Arduinolta

vastauksena lähetetyn verkkotunnuksen ja salasanan. Funktiosta palataan vasta, kun molemmat on luettu oikein, eli niiden ensimmäinen merkki on komentomerkki ’S’ verkkotunnukselle ja ’P’ salasanalle. Yhdistämistä ja tunnusten lukemista jatketaan, kunnes WLAN-yhteys on luotu. Kun yhteys on muodostettu, sen kadotessa tai ESP:n käynnistyessä uudelleen, ESP yhdistyy automaattisesti takaisin samaan verkkoon. [27]

5. KOMMUNIKAATIO IOT-ALUSTAN KANSSA

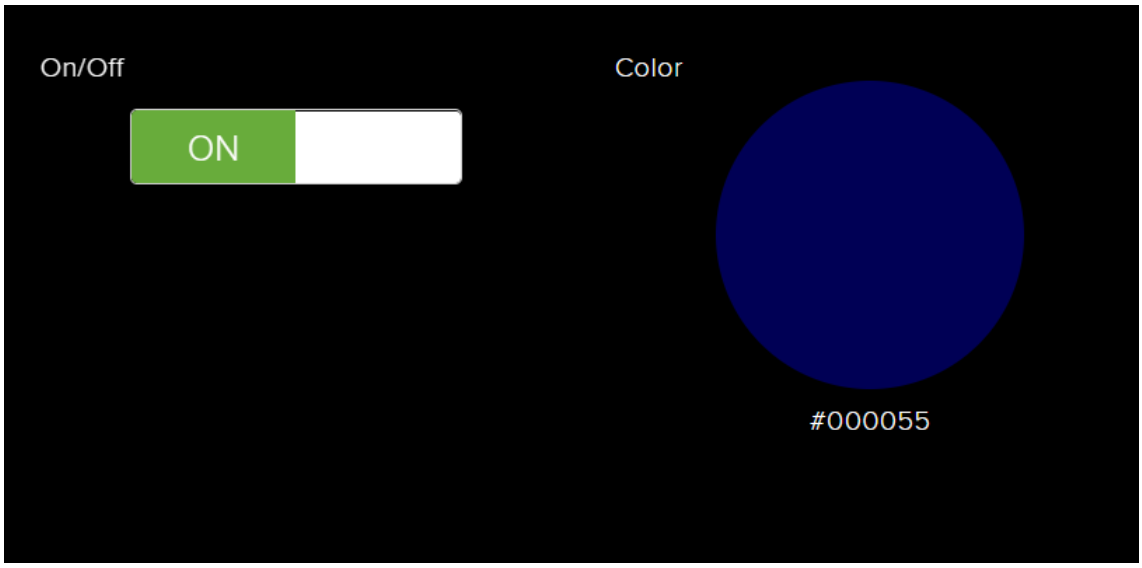
IoT-laitteen kommunikointi IoT-palvelimen kanssa on hyvin keskeinen prosessi laitteen toteutuksessa. Ilman datan luotettavaa välitystä molempiin suuntiin on IoT-laitteen mahdotonta toimia. Haluttu luotettavuuden taso riippuu IoT-laitteen käyttötarkoituksesta. Esimerkiksi jatkuvasti dataa lähettävän lämpötilasensorin ei ole välttämätöntä lähettää jokaista datapistettä virheettä palvelimelle, mutta vaikkapa terveydenhuoltoon liittyvän laitteen täytyy taata jokaisen datapisteen välittyminen.

5.1 IoT-alustan toiminta

Adafruit IO on valittu tämän laitteen IoT-alustaksi, sillä se tukee palvelutasoa (engl. Quality of Service, QoS) -tasoa 1, jonka merkitys selitetään luvussa 5.2. Lisäksi sen dashboard on selkeä ja helppokäyttöinen, sen käyttö ei vaadi omia palvelimia ja sillä on omat kirjastot Arduino-ympäristöä varten [28]. Tässä sovelluksessa käytetään Message Queuing Telemetry Transport -protokollaa (MQTT, kevyt viestiprotokolla) ja AdafruitMQTT-kirjastoa Adafruit IO:n kanssa kommunikointiin. Adafruit tarjoaa myös AdafruitIO-kirjaston, joka on helppokäyttöisempi ja tarjoaa enemmän toiminnallisuuksia, mutta AdafruitMQTT:tä käyttämällä MQTT-toteutus on selkeämmin esillä koodissa [29, 30].

Adafruit IO:ssa data on jaettu syötteisiin (engl. feed). Syöte on Adafruit IO:ssa käytetty termi, joka toimii MQTT-protokollan aiheena ja sisältää lisäksi metadataa, kuten datan julkisuus ja kuvaus [31]. Syötteisiin lähetetyn datan tyyppi voi olla numeerinen, merkkijono tai JavaScript Object Notation (JSON) [28]. Tässä sovelluksessa käytettyjä syötteitä on kaksi. Toinen sisältää led-nauhan päälle/pois päältä -tilan merkkijonona ”ON” tai ”OFF” ja toinen led-nauhan värin HEX-värikoodina. HEX-värikoodi on merkkijono, joka on muotoa ”#RRGGBB”, missä kullekin värille on varattu kaksi heksadesimaalimerkkiä (R - punainen, G - vihreä, B - sininen) eli kahdeksan bittiä [32].

Syötteille voidaan lähettää dataa verkkosivulta, IoT-laitteelta tai Adafruit IO:n dashboardeilta. Dashboard on web-sovellus, jonka kautta voidaan visualisoida ja lähettää syötteelle dataa. Dashboardille voidaan lisätä osioita (engl. block), jotka toimivat tietyn syötteen sisäänmenoina, ulostuloina tai molempina. [33] Dashboard soveltuu hyvin prototyyppien tai harrastelijoiden laitteiden ohjaamiseen, sillä sitä käytettäessä ei tarvitse kehittää erillistä verkkosivua tai muuta ohjelmistoa, jolla syötteitä ohjataan. Vaikka dashboardeja voidaan luoda koneellisesti, suuremman mittakaavan tuotannoissa tulisi luoda, ohjata ja visualisoida syötteitä siihen tarkoitetulta verkkosivulta, jos halutaan parantaa käyttäjäystävällisyyttä ja piilottaa Adafruit-toteutus. Syötteiden verkkosivuohjausta tarvitaan myös, jos niiden hallitsemisessa on käytettävä logiikkaa, jota ei voida toteuttaa pelkästään dashboardin osioilla.



Kuva 7. *AdafruitIO:n dashboard led-valon ohjaamiseen*

Koska tässä työssä on tarkoitus rakentaa yksittäinen laite, valitaan dashboard sen ohjaamiseen. Sovelluksessa on kaksi syötettä, joten dashboardilla on myös kaksi niitä vastaavaa osiota, led-nauhan tila ja väri, jotka näkyvät kuvassa 7.

5.2 MQTT-protokolla

MQTT on kevyeksi, helppokäyttöiseksi ja luotettavaksi suunniteltu viestiprotokolla, joka perustuu publish-subscribe-malliin. MQTT:n käyttökohteita ovat muun muassa sovellukset, joissa verkkoyhteyden laatu tai nopeus on huono, sekä sulautetut järjestelmät, joilla on rajoitettu prosessointi- tai muistikapasiteetti. [34] MQTT toimii Transfer Control Protocol/Internet Protocol (TCP/IP, lähetysohjausprotokolla/internet-protokolla) -protokollapinon päällä. [35]

MQTT-viestiprotokolla sopii hyvin IoT-sovelluksiin, sillä sitä käytettäessä IoT-laitteen ei tarvitse pollata palvelinta, vaan palvelin lähettää viestit automaattisesti niistä kiinnostuneille laitteille. Tämä säästää laskentatehoa vähentämällä epäoleellisen verkkoviestinnän määrää. MQTT:ssä laitteet jaetaan välittäjiin (engl. broker) ja asiakkaisiin (engl. client). Välittäjän data on jaettu aiheisiin (engl. topic), joita asiakkaat voivat tilata (engl. subscribe) ja joihin ne voivat julkaista (engl. publish). Kun johonkin aiheeseen julkaistaan dataa, välittäjä lähettää sen kaikille aiheen tilanneille asiakkaille. [1]

MQTT:ssä asiakkaat siis eivät ole suoraan yhteydessä toisiinsa, vaan vain välittäjään, ja sama asiakas voi tilata ja julkaista useisiin aiheisiin. Tämä siirtää vastuun viestien välittymisestä oikeille laitteille välittäjälle, mikä säästää asiakkaiden rajoitettua laskentatehoa, etenkin jos asiakkaita on useita.

Aiheet erotetaan toisistaan hierarkkisen nimeämiskäytännön avulla. Nimet ovat merkkijonoja, joiden eri hierarkiatasoja erotetaan /-merkillä. Esimerkiksi Adafruit IO:n nimeämiskäytännössä, muotoa ”<käyttäjänimi>/feeds/<syötteen nimi>”, käyttäjänimi on hierarkiatasossa ylimpänä ja syötteen nimi alimpana [28]. Villikortit (engl. wildcard) ovat merkkejä, joilla asiakas voi tilata useamman aiheen kerrallaan. Villikorttimerkki + on yksitasoinen, ja sen avulla tilataan vain yhden hierarkiatason aiheet asettamalla se kyseisen tason kohdalle. Monitasoista villikorttia kuvataan #-merkillä, ja sen avulla tilataan kaikki aiheet, jotka ovat hierarkiatasolla, johon merkki asetetaan, ja tämän tason alapuolella. [34]

MQTT-protokolla määrittelee QoS:n, joka kuvaa viestien välityksen luotettavuutta. QoS:lle on määritelty kolme arvoa [34] :

- QoS 0 takaa ”korkeintaan kerran” -tason luotettavuuden, eli viestin saapuminen välittäjälle ei ole taattua ja se saattaa duplikoitua.
- QoS 1 takaa ”ainakin kerran” -tason luotettavuuden, eli viesti saapuu varmasti välittäjälle, mutta se saattaa duplikoitua.
- QoS 2 takaa ”tasan kerran” -tason luotettavuuden, eli viesti saapuu varmasti ja vain yhden kerran välittäjälle.

MQTT-yhteys aloitetaan asiakkaan lähettämällä CONNECT-paketilla, jonka välittäjä tarkastaa ja todentaa, ja vastaa sitten CONACK-paketilla. CONNECT-paketti sisältää ClientID:n, CleanSession-lipun, käyttäjänimen, salasanan, käytetyn MQTT-protokollaversion, KeepAlive-kentän sekä last will -ominaisuuteen liittyvät arvot Will, WillQoS, WillRetain, WillTopic ja WillMessage. CONACK-paketti sisältää SessionPresent-lipun, joka kertoo, onko asiakkaalla istunto kesken, vai aloitetaanko uusi istunto ja ReturnCode-kentän, jonka arvo on luku väliltä 0-5. Luku 0 tarkoittaa yhteyden luomisen onnistumista ja luvut 1-5 vastaavat jokainen jotakin virheilmoitusta. [35]

ClientID on asiakkaan tunnistamiseen käytetty merkkijono, jonka tulee olla uniikki jokaiselle asiakkaalle, sillä sen perusteella välittäjä pitää kirjaa asiakkaan istunnosta. ClientID voi olla myös tyhjä, jolloin välittäjä luo sen asiakkaan puolesta. Jos CleanSession-lipun arvoksi asetetaan 1, eli tosi, tallentaa välittäjä asiakkaan istunnon tiedot vain yhteyden ajaksi. Jos taas arvoksi asetetaan 0, epätosi, se tarkoittaa asiakkaan haluavan, että välittäjä tallentaa istuntotiedot, kuten tilaukset ja tilattuihin aiheisiin saapuneet viestit, joilla on QoS-taso 1 tai 2. Tällöin nämä ovat suoraan asiakkaan käytössä yhteyden katkeamisen ja sen uudelleen luomisen jälkeen. [35] AdafruitMQTT-kirjaston oletusarvo CleanSession lipulle on 1 [36].

Jos asiakas haluaa asettaa itselleen käyttäjänimen ja salasanan todennusta varten, on sen asetettava UserName- ja Password-liput arvoon 1, ja annettava saman nimisille kentille palvelimen tunnistamat arvot. KeepAlive-kentän arvo määrittelee sen ajan sekunneissa, kuinka kauan yhteyttä pidetään auki, jos asiakas ei lähetä palvelimelle viestejä. Jos asi-

akkaan ei tarvitse tämän ajan sisällä lähettää viestejä, on sen mahdollista lähettää PING-REQ-paketti, josta välittäjä tietää yhteyden olevan käytössä ja johon se vastaa PING-RESP-paketilla, jolloin myös asiakas tietää yhteyden olevan käytössä. KeepAlive-arvon ollessa 0, tämä ominaisuus on poissa käytöstä. [35] AdafruitMQTT-kirjaston oletusarvo KeepAlivelle on 300 s [36].

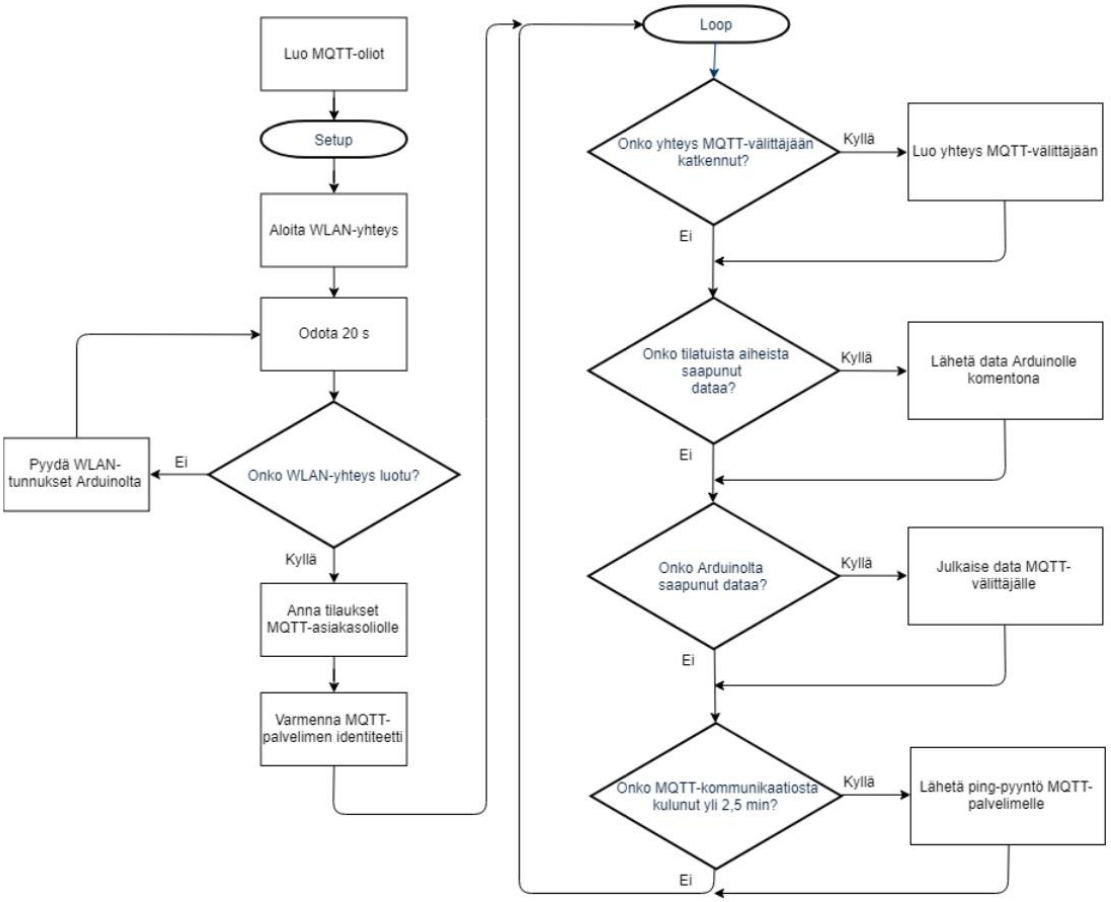
MQTT:n last will -ominaisuus aktivoidaan asettamalla Will-lippu arvoon 1, jolloin asiakkaan antama WillMessage-kentän arvo julkaistaan WillTopic-kentän sisältämässä aiheessa palvelimen toimesta käyttämällä QoS-arvoa WillQoS, jos yhteys asiakkaaseen menetetään tai katkaistaan. Asiakas voi siis ”lähettää” palvelimelle viestin yhteyden katketessa. [35]

Asiakas ilmoittaa halunsa tilata jokin aihe SUBSCRIBE-paketilla, joka lähetetään aina QoS-tasolla 1, ja johon välittäjä vastaa SUBACK-paketilla. SUBSRIBE-paketti sisältää listan asiakkaan tilaamista aiheista, sekä näiden halutut QoS-tasot. Välittäjä voi asettaa QoS-tason alemmas kuin käyttäjä pyytää, jos sitä ei pystytä takaamaan. SUBACK-paketti sisältää tilauksille asetetut QoS-tasot. [34]

Kun asiakas haluaa julkaista dataa johonkin aiheeseen, se lähettää PUBLISH-paketin, johon välittäjä vastaa paketin QoS-tasosta riippuen. QoS-tasolla 0 pakettiin ei vastata, tasolla 1 vastaus on PUBACK-paketti ja tasolla 2 vastaus on PUBREC-paketti. PUBLISH-paketti sisältää aiheen nimen, johon dataa julkaistaan, paketin QoS-tason sekä itse julkaistavan datan. [34]

5.3 MQTT-protokollan käyttö

AdafruitMQTT-kirjaston avulla MQTT-kommunikaatio on yksinkertaista, sillä kirjasto suorittaa varsinaisen yhteyden muodostamisen sekä pakettien kokoamisen, lähettämisen, vastaanottamisen ja purkamisen. Tässä laitteessa ESP:n tehtävänä on ylläpitää internetyhteyttä, välittää MQTT-välittäjältä saatu data Arduinolle ja Arduinolta saatu data MQTT-välittäjälle, ja tämän prosessin vaiheet on esitetty kuvassa 8.



Kuva 8. *ESP:n ohjelman toiminta vuokaaviona*

AdafruitMQTT-kirjaston avulla yhdistäminen välittäjään ohjelmassa 10 tapahtuu seuraavasti. Ensin luodaan ESP8266WiFi-kirjaston WiFiClientSecure-olio, jonka avulla ESP voi lähettää, vastaanottaa ja prosessoida palvelinten lähettämää dataa [27]. Sitten luodaan asiakasolio (Adafruit_MQTT_Client), joka käyttää edellä luotua WiFiClientSecure -oliota MQTT-palvelimen kanssa kommunikointiin. Tälle oliolle annetaan myös MQTT-palvelimen URL-osoite, portti, johon yhteys luodaan, käyttäjänimi sekä salasana. Käyttäjänimi on Adafruit IO -palveluun rekisteröityessä luotu käyttäjänimi, ja salasana on Adafruit IO:n luoma AIO KEY, joka on uniikki jokaiselle käyttäjälle. Kutsumalla tämän olion connect-metodia, AdafruitMQTT-kirjasto luo yhteyden palvelimeen ja palauttaa luvun 0, mikäli tämä onnistuu. Jos palautettu luku on muu kuin 0, yhteyden luomisessa on tapahtunut virhe, ja kyseinen luku vastaa jotakin virhekoodia. [37]

```
//Vakioden määrittely ja yhteyden luomiseen tarvittavien olioiden luo
//minen ohjelman alussa:
#define AIO_SERVER      "io.adafruit.com"
#define AIO_SERVERPORT  8883
#define AIO_USERNAME    "<Adafruit IO -käyttäjänimi>"
#define AIO_KEY         "<Adafruit IO:n tarjoama avain>" //MQTT-sala
                                                         //sana

WiFiClientSecure client;
Adafruit_MQTT_Client mqtt(&client, AIO_SERVER, AIO_SERVERPORT,
AIO_USERNAME, AIO_KEY); //asiakasolion luominen

--- Välittäjään yhdistäminen loop-osiossa: ---
if ( !mqtt.connected() ){
    yhdistaMQTT();
}

--- yhdistaMQTT-funktion määrittely ---
void yhdistaMQTT(){
  while ( mqtt.connect() != 0{
    mqtt.disconnect();
    delay(5000); //Odota 5 s ennen uutta yhdistysyritystä
  }
}
```

Ohjelma 10. *MQTT-yhteyden luominen AdafruitIO-palvelimeen*

ESP:n ohjelman loop-osiossa kutsutaan yhdistaMQTT-funktiota, jos yhteys MQTT-palvelimeen on katkennut. YhdistaMQTT-funktio kutsuu asiakasolion connect-metodia, kunnes se palauttaa luvun 0, eli yhteys on muodostettu onnistuneesti.

Aiheisiin julkaiseminen AdafruitMQTT-kirjaston avulla vaatii julkaisuolion (Adafruit_MQTT_Publish) luomista jokaiselle aiheelle, johon halutaan julkaista. Tässä tapauksessa aiheita on kaksi, ja ne on nimetty onoff (led-nauhan tila) ja color (led-nauhan väri), joten molemmille luodaan omat oliot. Olioille annetaan lisäksi aiemmin luotu asiakasolio, joka avulla se kommunikoi välittäjän kanssa, ja molempien QoS-tasoksi asetetaan 1, sillä kaikki tehdyt muutokset led-nauhan tilassa halutaan julkaista vähintään kerran.

```
//Julkaisuolioiden luominen ohjelman alussa:
Adafruit_MQTT_Publish onoffOut = Adafruit_MQTT_Publish( &mqtt,
AIO_USERNAME "/feeds/onoff", MQTT_QOS_1 );
Adafruit_MQTT_Publish colorOut = Adafruit_MQTT_Publish( &mqtt,
AIO_USERNAME "/feeds/color", MQTT_QOS_1 );

//Arduinolta saadun komennon lukeminen, ja sen perusteella julkaisemi
//nen loop-osiossa:
char* komento;
komento = lueKomento(&Serial,cd);

if (komento != NULL){

  if (komento[0] == 'L'){
    //Julkaistaan onoff-syötteeseen,joten julkaistaan komennon para
    //metrimerkkijono sellaisenaan
    ++komento;
    while ( !onoffOut.publish(komento) ){
      //Julkaisu epäonnistui, yritetään uudelleen
      delay(2000);
      yhdistaMQTT();
    }

  } else if ( komento[0] == 'C' ){
      //Julkaistaan color-syötteeseen, joten vaihdetaan saadun
      //komentomerkkijonon ensimmäinen merkki merkkiin #, ja
      //julkaistaan se
      komento[0] = '#';
      while ( !colorOut.publish(komento) ){
        delay(2000);
        yhdistaMQTT();
      }
  }

    viimViesti = millis();
}
```

Ohjelma 11. *Julkaiseminen AdafruitIO-palvelimelle*

Ohjelmassa 11 esitetty varsinainen julkaisu tapahtuu ohjelman loop-osiossa, kun ESP on saanut Arduinolta komennon sarjaliikenneyhteydellä. Aihe, johon julkaistaan, valitaan komennon ensimmäisen merkin perusteella. Julkaisuun käytetään publish-metodia, jolle annetaan parametrinä julkaistava data. Onoff-syötteen tapauksessa komennon parametrimerkkijono on suoraan syötteen hyväksymässä muodossa, mutta color-syöte vaatii #-merkin lisäämistä merkkijonon alkuun. Koska julkaisuille on määritelty QoS-taso 1, palauttaa publish-metodi arvon epätosi, mikäli julkaisu epäonnistuu. Jos näin käy odotetaan 2 s, ja yritetään julkaisua sitten uudestaan, kunnes se onnistuu.

Aiheiden tilaaminen AdafruitMQTT-kirjaston avulla vaatii tilausolion (Adafruit_MQTT_Subscribe) luomista jokaiselle aiheelle, joka halutaan tilata. Tilausolio käyttää julkaisuolion tavoin asiakasoliota välittäjän kanssa kommunikointiin. Tilausolion QoS-tasoksi asetetaan 1.

```
--- Tilausolioiden luominen ohjelman alussa: ---
Adafruit_MQTT_Subscribe onoffIn = Adafruit_MQTT_Subscribe(&mqtt, AIO_USERNAME "/feeds/onoff", MQTT_QOS_1);
Adafruit_MQTT_Subscribe colorIn = Adafruit_MQTT_Subscribe(&mqtt, AIO_USERNAME "/feeds/color", MQTT_QOS_1);

--- Aiheiden tilaaminen setup-osiossa: ---
mqtt.subscribe(&onoffIn);
mqtt.subscribe(&colorIn);

//Tilatuista aiheista saapuneen datan lukeminen loop-osiossa:
//Luo tilausolio-osoitin, jolla tarkistetaan, mihin tilaukseen on saapunut dataa
Adafruit_MQTT_Subscribe *subscription;

//Silmukka, joka lukee tilauksista saapunutta dataa 5000 ms
while ( (subscription = mqtt.readSubscription(5000)) ) {
  if ( subscription == &onoffIn ){
    //Dataa saapunut onoff-syötteeseen
    sprintf(ledState, "%s", (char *)onoffIn.lastread);
    lahetaKomento(&Serial,'L',ledState); //Lähetä data komentona Arduinolle
    viimViesti = millis();

  } else if ( subscription == &colorIn ){
    //Dataa saapunut color-syötteeseen
    //Lue saapunut data väliaikaiseen muuttujaan ja poista ensimmäinen
    //merkki (#)
    char tmp[7];
    sprintf(tmp, "%s", (char *)colorIn.lastread);
    char *ptr = tmp;
    ++ptr;

    lahetaKomento(&Serial,'C',ptr); //Lähetä data komentona Arduinolle
    viimViesti = millis();
  }
}
```

Ohjelma 12. *Tilatuista syötteistä saapuneen datan lukeminen*

Aiheiden tilaaminen tapahtuu setup-osiossa asiakasolion subscibe-metodilla, joka lisää tilausoliot sen tilattujen syötteiden listalle, ja tilaa listan automaattisesti yhteyden luomisen jälkeen. Tilatuista aiheista saatua dataa luetaan ohjelmassa 12 asiakasolion readSubscription-metodilla, joka palauttaa osoittimen tilausolioon, jonka aiheeseen saapunut data kuuluu, mikäli dataa saapuu. Metodia kutsutaan loop-osiossa, ja sen palauttaman osoittimen perusteella valitaan, mikä komentomerkki asetetaan Arduinolle lähetettävään komentoon. Saapunut data luetaan tilausolion lastread-metodilla, ja asetetaan Arduinolle lähetettävän komennon parametrimerkkijonoksi. Mikäli saatu data on color-syötteestä, sen ensimmäinen merkki (#) poistetaan.

```
  --- Ping-viestin lähetys loop-osiossa: ---
if ( (millis() - viimViesti) > 150000){
  DEBUGPRINTLN("Ping!");
  if( mqtt.ping()) {
    viimViesti = millis();
  } else {
    mqtt.disconnect();
  }
}
```

Ohjelma 13. *Ping-viestin lähetys*

MQTT-yhteyden ylläpitämiseksi ESP lähettää PINGREQ-viestin 2,5 minuuttia viimeisimmän välittäjälle lähetetyn viestin lähettämisen jälkeen ohjelmassa 13 asiakasolion ping-metodilla, joka palauttaa arvon tosi, mikäli välittäjä vastaa PINGRESP-viestillä. Tässä tapauksessa tallennetaan viestin saapumisaika viimViesti-muuttujaan. Tähän muuttujaan tallennetaan aika aina viestejä lähettäessä ja vastaanottaessa, jotta huomataan, kun viimeisestä viestistä on kulunut yli 2,5 min. Jos ping palauttaa arvon epätosi, MQTT-yhteys lopetetaan, jolloin yhteys luodaan uudelleen loop-osion alussa.

5.4 HTTP-kommunikaatio

Adafruit IO tarjoaa myös hypertekstin siirtoprotokolla (engl. Hypertext Transfer Protocol, HTTP) ohjelmointirajapinnan (engl. Application Programming Interfacen, API), jonka avulla voidaan hallita ohjelmallisesti muun muassa syötteitä, niiden dataa, dashboardeja ja triggereitä. Tätä API:a käytetään lähettämällä HTTP-pyyntö palvelimelle, joka suorittaa halutun toiminnon ja vastaa sitten dokumentaation osoittamalla status-koodilla sekä viestillä. [15]

HTTP API sopii myös useamman datapisteen hakemiseen kerralla, kun hakutaajuus ei ole tiheä, sillä joka kerta dataa haettaessa joudutaan ensin kysymään sitä palvelimelta. Jos tarvitaan reaaliaikaisia päivityksiä syötteen tilasta, on MQTT parempi vaihtoehto. Tässä laitteessa ei siis tarvita HTTP:tä tietoliikenteeseen IoT-alustan kanssa.

6. TIETOTURVARISKIT JA NIIDEN VÄLTTÄMINEN

Tietoturva on yksi tärkeimmistä osa-alueista IoT-laitteiden kehittämisessä ja toiminnassa. Internet-yhteydellä varustettujen laitteiden määrän kasvaessa ja niiden sovellusalueiden laajentuessa, hyökkääjien mahdollisuudet kasvavat etenkin, jos tietoturvaan ei kiinnitetä erityistä huomiota. Harva suojaus on murtamaton, varsinkin jos sen murtamisesta on saatavissa merkittävää hyötyä murtajalle. IoT-tietoturvan suurimmat haasteet liittyvät heterogeenisen laitekannan kommunikointiin keskenään, IoT-protokollien turvallisuuden parantamiseen, laitteiden kehittämiseen turvallisiksi alusta alkaen sekä vaarantuneiden laitteiden tunnistamiseen. [38]

6.1 IoT led -valon tietoturvariskit

Yksittäisiin prototyyppeihin tai harrastelijoiden laitteisiin hyökkääminen ei ole yhtä arvokasta hyökkääjälle kuin teollisesti valmistettuihin laitteisiin hyökkääminen, sillä kohdistamalla hyökkäys teollisesti valmistettuihin laitteisiin voidaan saada pääsy useisiin samanlaisiin laitteisiin. Yksittäisten prototyyppien tapauksessa hyökkäys voidaan toisaalta kohdistaa esimerkiksi käytettyihin kirjastoihin, koska niitä käytetään useissa laitteissa. Tämän laitteen tapauksessa käytetyt kirjastot oletetaan tarpeeksi luotettaviksi, sillä jos näin ei tehtäisi, tulisi kirjastot tarkastaa kokonaan haavoittuvuuksien varalta tai kirjoittaa kokonaan uudet kirjastot. Tähän vaadittava työmäärä on kuitenkin hyötyyn nähden liian suuri.

Yksi tämän laitteen tietoturvariski on sekä WLAN, että AdafruitIO:n salasanojen ja käyttäjätunnusten tallentaminen selvätekstinä, sillä niiden avulla hyökkääjä pääsee mahdollisesti käsiksi muihin laitteisiin ja pystyy tarkkailemaan langatonta lähiverkkoliikennettä [39]. Helpoin tapa päästä käsiksi näihin tunnuksiin on fyysisesti laitteen kautta. Laitteita ei kuitenkaan valmisteta useita myyntiin ja laite sijaitsee käyttäjän kotona, minkä takia tässä laitteessa riski on pieni. Tunnusten lisäksi tähän laitteeseen hyökkääminen antaa hyökkääjälle mahdollisuuden vain led-valon hallitsemiseen, mikä ei ole erityisen hyödyllistä tai haitallista, mutta se on kuitenkin pyrittävä estämään.

Laitteen fyysinen tietoturvariski voitaisiin minimoida AtMega328p:n lukkobittien avulla. Kun nämä lukkobitit asetetaan tiettyihin arvoihin, voidaan joko vain laitteen uudelleen ohjelmointi tai myös ohjelman ulkoinen lukeminen estää [6]. ESP8266EX ei kuitenkaan tarjoa mahdollisuutta salata ohjelmaa tai estää ohjelman ylikirjoittamista, sillä se tallennetaan mikrokontrollerin ulkoiseen flash-muistiin [24]. Mikäli ohjelma halutaan salata, on WLAN-moduuli vaihdettava esimerkiksi ESP32-malliin, joka tukee flash-muistin salausta ja uudelleen ohjelmoinnin estämistä [40]. Tämä kuitenkin estäisi myös ohjelmiston

ja seuraavassa luvussa mainitun SHA1-tiivisteen päivittämisen. Laite on siis avoin fyysiselle hyökkäykselle, mutta hyöty mahdollisuudesta päivittää ohjelmistoa on suurempi kuin tämä riski.

Toinen laitteeseen kohdistuva tietoturvauhka on sen tietoliikenteen salakuuntelu, jonka riski on suurempi kuin sen fyysinen väärinkäyttö, sillä se voidaan suorittaa internetin yli esimerkiksi IoT-alustan tietoliikennettä seuraamalla. Tämä uhka on myös fyysistä uhkaa käytännöllisemmin estettävissä salaamalla tietoliikenne.

6.2 Tietoliikenteen salaus

Koska MQTT-paketit lähetetään selvätekstinä, niiden sisältämä yksityinen informaatio, kuten salasanat, on suojattava käyttämällä salausprotokollaa. Adafruit IO tarjoaa tuen yleisesti käytetylle Transport Layer Security -protokollalle (TLS, kuljetuskerroksen suojaus), jota käytetään tässä laitteessa MQTT-pakettien salaukseen [41]. TLS:n aiempi versio on nimeltään Secure Socets Layer (SSL, kuljetuskerroksen suojaus), ja puhekielessä näitä termejä käytettään yleisesti synonyymeinä. TLS:n uusin versio on TLS 1.2, joka on julkaistu vuonna 2008. [42] ESP8266WiFi-kirjasto käyttää TLS-suojaukseen axTLS-kirjastoa [43]. AxTLS-kirjasto tukee TLS-versiota 1.2 ja se on suunniteltu käytettäväksi laitteissa, joissa on rajoitettu muistikapasiteetti [44].

TLS-suojaus perustuu julkisen avaimen salaukseen, jossa käytetään kahta avainta, julkista ja salaista. TLS:n kättelyprotokolla välittää julkisen avaimen palvelimelta asiakkaalle yhdessä palvelimen identifioimiseen käytetyn varmenteen kanssa. Julkisella avaimella salatut viestit voidaan purkaa vain salaisen avaimen avulla, joka on vain palvelimen käytössä. Julkista avainta käytetään siis vain salaamiseen ja salaista avainta vain salauksen purkamiseen. Kun asiakas on saanut palvelimen varmenteen, se varmentaa palvelimen identiteetin varmenteen perusteella, mikäli se on tarpeellista. TLS-salaus toimii myös, vaikka palvelimen identiteetti jätettäisiin varmentamatta, mutta tällöin ei ole varmuutta palvelimen oikeasta identiteetistä. Kättelyprotokolla myös huolehtii, että asiakas ja palvelin käyttävät samaa suojauspakettia (engl. cipher suite), eli kokoelmaa salausalgoritmeja. Kun suojauspaketti on neuvoteltu, asiakas luo pääsalaisuuden, eli avaimen, jolla tuleva liikenne salataan. Pääsalaisuus lähetetään palvelimelle suojauspaketin määräämällä tavalla, julkisella avaimella salattuna. Nyt molemmilla osapuolilla on tiedossa pääsalaisuus, jota tarvitaan viestien salaukseen ja purkamiseen, sekä suojauspaketti, joka kertoo, miten viestit salataan ja puretaan. Näin on saatu aikaan salattu yhteys. [42]

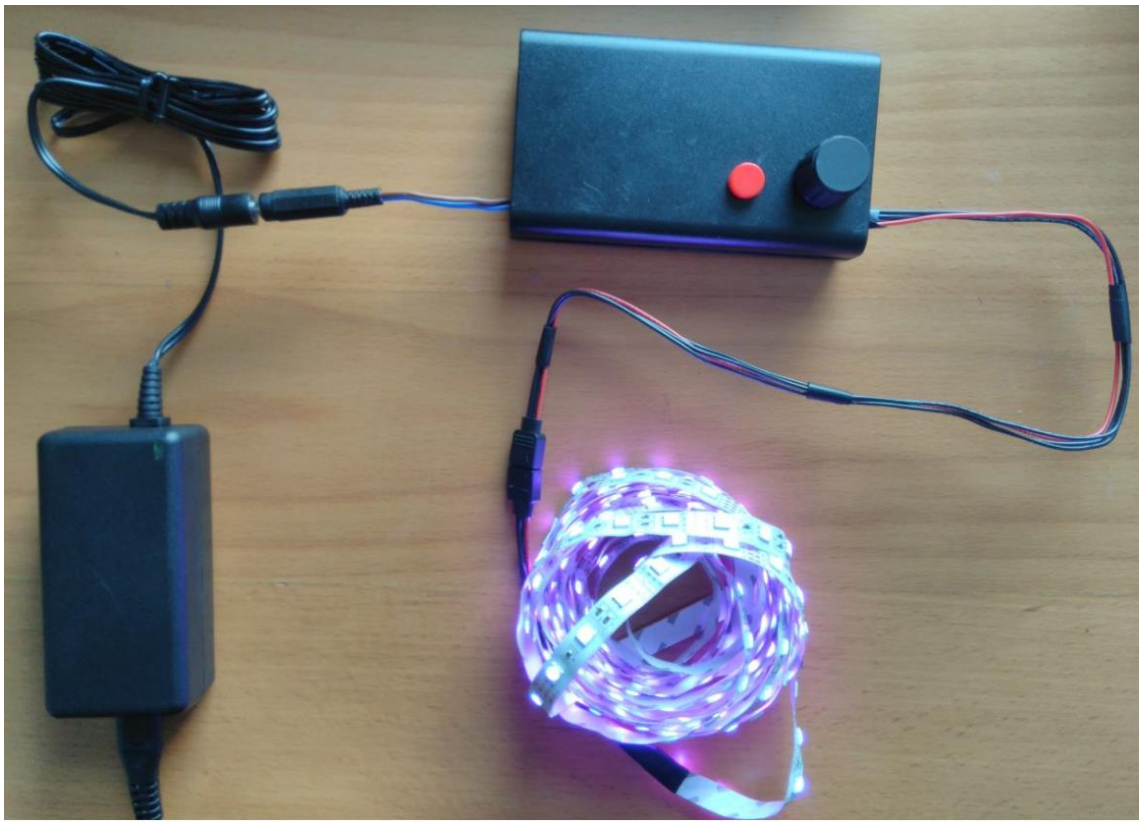
ESP8266WiFi-kirjastossa on TSL-suojattua yhteyttä varten luotava WiFiClientSecure-olio. Tämä olio toimii kuten Arduinon vakiokirjastoon perustuva WiFiClient-olio, mutta se käyttää TLS:ää palvelinyhteyden salaamiseen [27]. WiFiClientSecure-oliolla on lisäksi verify-metodi, jonka avulla palvelimen identiteetti voidaan varmistaa kättelyn aikana palvelimelta saadun varmenteen avulla. Verify-metodi tarvitsee saadun varmenteen lisäksi Secure Hash Algorithm (SHA, kryptografinen tiivistealgoritmi) -menetelmällä

tuotetun tiivisteen varmenteesta, jota verrataan saatuun varmenteeseen. Mikäli tiiviste on oikeaa muotoa ja täsmää varmenteeseen, palvelin katsotaan luotettavaksi, verify palauttaa arvon tosi ja ohjelman suoritus jatkuu normaalisti. Verifyn palauttaman arvon ollessa epätosi ESP resetoidaan. SHA-tiiviste on saatavilla AdafruitMQTT-kirjaston esimerkkikoodista, ja sen on kovakoodattava mikrokontrollerin ohjelmaan. [37] Kovakoodauksen haittapuoli tulee esille varmenteen vanhentuessa, jolloin tiiviste on vaihdettava ohjelmoimalla mikrokontrolleri uudelleen.

Laitteen suunnittelun ja rakentamisen jälkeen axTLS-kirjastoon perustuva WiFiClientSecure on merkitty ESP8266WiFi-kirjaston dokumentaatiossa vanhentuneeksi, ja vaikka sitä tuetaan edelleen, on sen päivittäminen lopetettu [27]. Tämä johtaa kasvaneeseen tietoturvariskiin, sillä mikäli tästä axTLS:ssään perustuvasta toteutuksesta löytyy haavoittuvuuksia, niitä ei pyritä korjaamaan. Laitten ohjelmisto tulee kuitenkin päivittää manuaalisesti haavoittuvuuksien löytyessä tai tasaisin väliajoin, joten tällaisessa tapauksessa ohjelmistoa voidaan muuttaa tukemaan uutta BearSSL-kirjastoon perustuvaa toteutusta pelkän kirjastopäivityksen sijasta.

7. YHTEENVETO

Työ onnistui johdannossa määriteltyjen vaatimusten mukaisesti. Sillä on kaksisuuntainen yhteys palvelimeen eli led-nauhan tilaa ja väriä voidaan vaihtaa sekä IoT-alustalta, että laitteen fyysiseltä käyttöliittymältä. Viive palvelimelta laitteelle on lähes reaaliaikainen, led-nauha valaisee tunnelmavalaistukselle sopivalla voimakkuudella ja laite toimii myös ilman internetyhteyttä.



Kuva 9. *Valmis IoT led -valo*

Kuvassa 9 on esitetty valmis IoT led -valo toiminnassa alhaisella kirkkaudella. Fyysisen käyttöliittymän, punainen painonappi ja potentiometri-säädin kotelossa, toteutus onnistui hyvin ja se toimii luotettavasti.

IoT led -valon keskeisimmät tekniset tiedot sekä toteutukseen käytetyt valmiit ohjelmisto- ja laitteistokomponentit on esitetty taulukossa 4.

Taulukko 5. *Yhteenveto IoT led -valon teknisistä tiedoista*

Valaisimen tyyppi	RGB led -nauha
Himmennysmetodi	PWM-himmennys
Jännitelähde	CPS-12030C8
Käyttöjännite	12 V
Maksimivirrankulutus ja -teho	2,5 A, 30 W
Teholähde	12 V 36 W hakkurijännitelähde
Ohjausyksikkö	Arduino Nano -kehitysalusta
WLAN-moduuli	ESP8266-01S
WLAN-taajuus	2,4 GHz
IoT-alusta	Adafruit IO
Tietoliikenneprotokollat	MQTT, TCP/IP, ad hoc-sarjaliikenne
Käytetyt kirjastot	Arduino standardikirjastot, ESP8266WiFi, AdafruitMQTT
Tietoturvariskit	Fyysinen peukalointi
Salausprotokolla	TLS

Joitakin ongelmia laitteen kehitysvaiheessa kuitenkin ilmeni. Piirilevyn valmistusprosessissa kaikki lakka ei irronnut kehityksen aikana liian lyhyen kehitysajan takia, mikä johti pidempään syövytysaikaan, jolloin osa vedoista syöpyi liian ohuiksi. Tämä voitiin kuitenkin korjata lisäämällä juotostinaa näille alueille.

Ohjelmiston kehittämisessä suurin ongelma oli palvelimelta saapuvien komentojen viive. Ensimmäisessä versiossa kaikki ohjaus tapahtui Arduinon kautta ja ESP:tä ohjattiin AT-komennoilla. Tästä seurasi kuitenkin se, että Arduinon muu toiminta lakkasi MQTT-viestien lukusilmukan aikana, mikä aiheutti viivettä sekä led-nauhan ohjaukseen, että viestien vastaanottamiseen. Led-nauhan ohjaukseen fyysiseltä käyttöliittymältä olisi voitu käyttää keskeytyksiä, mutta tämä lisäsi viivettä entisestään ja hankaloitti potentiometrin lukemista. Sekä laitteen toiminnan, että selkeän komponenttien vastuunjaon kannalta hyvä ratkaisu oli siis ohjelmoida ESP hallinnoimaan internetyhteyttä ja MQTT-kommunikaatiota itsenäisesti.

MQTT-kommunikaation toteuttaminen onnistui hyvin valmiin kirjaston avulla ja se toimii laitteessa luotettavasti. Ongelmia kuitenkin ilmeni sen tarkempaa toimintaa selvittäessä, sillä sen dokumentointi koostuu suurimmaksi osaksi esimerkeistä ja kommentointi on puutteellista. Toiminnan selvittämiseksi oli siis luettava lähdekoodia, joka on paikoitellen vaikeaselkoista.

Laitetta voitaisiin jatkokehittää esimerkiksi integroimalla se johonkin valmiiseen tai itse suunniteltuun kotiautomaatiojärjestelmään. Laitteen hyötysuhdetta voitaisiin myös parantaa vaihtamalla led-nauha hyötysuhteeltaan parempaan led-valaisimeen ja käyttämällä tarkoitusta varten suunniteltua virtalähdettä. Led-nauha voitaisiin myös vaihtaa digitaaliseen led-nauhaan, jolloin jokaista lediä voitaisiin ohjata yksitellen ja saada näin aikaan erilaisia valoefektejä.

LÄHTEET

[1] H.C. Hwang, J. Park, J.G. Shon, Design and Implementation of a Reliable Message Transmission System Based on MQTT Protocol in IoT, Wireless Personal Communications, Vol. 91, Iss. 4, 2016, pp. 1765-1777. Saatavissa (katsottu ID: Hwang2016): https://doi.org/10.1007/s11277-016-3398-2.

[2] The Internet of Things: A survey, in: Computer Networks, 2010, pp. 2787-2805.

[3] LD1117 Datasheet, STMicroelectronics, Litarature Number 2572, 2013, 45 p. Saatavissa: https://www.mouser.fi/datasheet/2/389/ld1117-974075.pdf.

[4] L78 Datasheet, STMicroelectronics, Literature Number DS0422, 2018, 55 p. Saatavissa: https://www.st.com/resource/en/datasheet/l78.pdf.

[5] H. Lee, I. Books24x7, Thermal Design: Heat Sinks, Thermoelectrics, Heat Pipes, Compact Heat Exchangers, and Solar Cells, Wiley, US, 2010, 630 p.

[6] ATmega328/P, Mircochip Technology Inc., 2018, 455 p. Saatavissa: http://ww1.microchip.com/downloads/en/devicedoc/atmega328_p%20avr%20mcu%20with%20picopower%20technology%20data%20sheet%2040001984a.pdf.

[7] C. Wang, ESP-01/07/12 Series Modules User's Manual, Ai-Thinker Inc, 2017, 23 p. Saatavissa: http://wiki.ai-thinker.com/_media/esp8266/esp8266_series_modules_user_manual_v1.1.pdf.

[8] H. Schutte, Bi-directional level shifter for
I²C-bus and other systems, Application Note AN97055, Philips Semiconductors, Eindhoven, 1997, 16 p. Saatavissa: https://cdn.sparkfun.com/tutorialimages/BD-LogicLevel-Converter/an97055.pdf.

[9] J. Bayle, C Programming for Arduino, 1st ed. Packt Publishing, GB, 2013, 610 p.

[10] S. Winder, I. Books24x7, Power Supplies for LED Driving, 1st ed. Newnes, US, 2008, 305 p.

[11] T. Cooper RGB LED Strips, Adafruit, nettisivu. Saatavissa (katsottu 30.10.2018): https://learn.adafruit.com/rgb-led-strips/schematic.

[12] K. Szholusha Dimming LEDs with pulse-width modulation, EE Times, nettisivu. Saatavissa (katsottu 15.12.2018): https://www.eetimes.com/document.asp?doc_id=1281013.

[13] Y. Lin, C. Leng, J. Wang, A simple LED driver with low dimming switch stress, International Journal of Circuit Theory and Applications, Vol. 44, Iss. 3, 2016, pp. 683-692.

[14] CPS Series, Cool Power Solutions Oy, 2018, 2 p. Saatavissa: https://docs.google.com/viewer?url=https://www.cps.fi/fi/productattachments/index/download?id=51.

[15] Adafruit IO REST API, Adafruit IO, nettisivu. Saatavissa (katsottu 30.10.2018): https://io.adafruit.com/api/docs/.

[16] What is Arduino? Arduino, nettisivu. Saatavissa (katsottu 16.8.2018): https://www.arduino.cc/en/Guide/Introduction.

[17] K. Palovuori, Mikrokontrollerit, Tampereen teknillinen yliopisto, Tampere, Luentomateriaali, 2018, 16 p.

[18] Arduino Nano, Arduino, nettisivu. Saatavissa (katsottu 26.11.2018): https://store.arduino.cc/arduino-nano.

[19] S. Dawoud, R. Peplow, Digital System Design : Use of Microcontroller, River Publishers, Aalborg, 2009, 570 p.

[20] F. Gembler, P. Stawicki, A. Rezeika, A. Saboor, M. Benda, I. Volosyak, Effects of monitor refresh rates on c-VEP BCIs, Lecture Notes in Computer Science, Springer, Cham, pp. 53-62.

[21] analogWrite(), Arduino, nettisivu. Saatavissa (katsottu 30.10.2018): https://www.arduino.cc/reference/en/language/functions/analog-io/analogwrite/.

[22] S. Muthu, F.J.P. Schuurmans, M.D. Pashley, Red, green, and blue LEDs for white light illumination, IEEE Journal of Selected Topics in Quantum Electronics, Vol. 8, Iss. 2, 2002, pp. 333-338.

[23] H. Barragán Wiring Environment, Wiring, nettisivu. Saatavissa (katsottu 12.12.2018): http://wiring.org.co/faq.html.

[24] ESP8266EX Datasheet, Espressif Systems, 2016, 24 p. Saatavissa: https://www.espressif.com/en/support/download/documents?keys=&field_type_tid%5B%5D=14.

[25] ESP8266 AT Instruction Set, Espressif, 2016, 64 p. Saatavissa: https://www.espressif.com/en/support/download/documents?keys=&field_type_tid%5B%5D=14.

[26] Arduino core for ESP8266 WiFi chip, Github, nettisivu. Saatavissa (katsottu 15.8.2018): https://github.com/esp8266/Arduino.

[27] I. Grokhotkov ESP8266WiFi library, arduino-esp8266.readthedocs.io, nettisivu. Saatavissa (katsottu 16.8.2018): https://arduino-esp8266.readthedocs.io/en/latest/esp8266wifi/readme.html.

[28] J. Cooper, Adafruit IO, Adafruit, 2018, 24 p. Saatavissa: https://cdn-learn.adafruit.com/downloads/pdf/adafruit-io.pdf.

[29] T. Treece Adafruit IO Arduino Library, Adafruit, nettisivu. Saatavissa (katsottu 20.10.2018): https://adafruit-io-arduino.readthedocs.io/en/latest/index.html.

[30] Adafruit Industries Adafruit MQTT Library, Github, nettisivu. Saatavissa (katsottu 25.10.2018): https://github.com/adafruit/Adafruit_MQTT_Library.

[31] T. Treece Adafruit IO Basics: Feeds, Adafruit, nettisivu. Saatavissa (katsottu 29.10.2018): https://learn.adafruit.com/adafruit-io-basics-feeds.

[32] Colors HEX, W3Schools, nettisivu. Saatavissa (katsottu 27.11.2018): https://www.w3schools.com/colors/colors_hexadecimal.asp.

[33] T. Treece Adafruit IO Basics: Dashboards, Adafruit, nettisivu. Saatavissa (katsottu 26.10.2018): https://learn.adafruit.com/adafruit-io-basics-dashboards.

[34] ISO/IEC PRF 20922, MQTT Version 3.1.1, OASIS, 2014, 73 p.

[35] G.C. Hillar, MQTT Essentials - A Lightweight IoT Protocol, Packt Publishing Ltd, Birmingham, 2017, 273 p.

[36] Adafruit Industries Adafruit_MQTT.h, Github, nettisivu. Saatavissa (katsottu 26.10.2018): https://github.com/adafruit/Adafruit_MQTT_Library/blob/master/Adafruit_MQTT.h.

[37] T. DiCola, Adafruit Industries, Adafruit MQTT Library ESP8266 Adafruit IO SSL/TLS example, Github, nettisivu. Saatavissa (katsottu 29.10.2018): https://github.com/adafruit/Adafruit_MQTT_Library/blob/master/examples/adafruitio_secure_esp8266/adafruitio_secure_esp8266.ino.

[38] R. Roman-Castro, J. Lopez, S. Gritzalis, Evolution and Trends in IoT Security, Computer, Vol. 51, Iss. 7, 2018, pp. 16-25.

[39] C. Agbeboaye, F.O. Akpojedje, J. Okoekhian, SECURITY THREATS ANALYSIS OF WIRELESS LOCAL AREA NETWORK Compusoft, Vol. 7, Iss. 6, 2018, pp. 2773-2779.

[40] Flash Encryption, Espressif Systems, nettisivu. Saatavissa (katsottu 4.12.2018): https://docs.espressif.com/projects/esp-idf/en/latest/security/flash-encryption.html.

[41] T. Treece IoT Security: Connecting Your ESP8266 to Adafruit IO with SSL/TLS, Adafruit IO, nettisivu. Saatavissa (katsottu 29.10.2018): https://io.adafruit.com/blog/security/2016/07/05/adafruit-io-security-esp8266/.

[42] S. Turner, Transport Layer Security, IEEE Internet Computing, Vol. 18, Iss. 6, 2014, pp. 60-63. https://ieeexplore-ieee-org.libproxy.tut.fi/document/6938667.

[43] I. Grokhotkov Client Secure Class, arduino-esp8266.readthedocs.io, nettisivu. Saatavissa (katsottu 29.8.2018): https://arduino-esp8266.readthedocs.io/en/latest/esp8266wifi/client-secure-class.html.

[44] C. Hamilton-Rich axTLS Embedded SSL, axTLS, nettisivu. Saatavissa (katsottu 29.8.2018): http://axtls.sourceforge.net/index.htm.