Miika Sinervä

Mittaustiedon kerääminen pilvitietokantaan Raspberry Pi:tä ja MQTT-protokollaa hyödyntäen

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Sähkö- ja automaatiotekniikka

Insinöörityö

30.4.2025

Tiivistelmä

Tekijä:	Miika Sinervä
Otsikko:	Mittaustiedon kerääminen pilvitieto-kantaan Raspberry Pi:tä ja MQTT-protokollaa hyödyntäen
Sivumäärä:	37 sivua + 2 liitettä
Aika:	30.4.2025
Tutkinto:	Insinööri (AMK)
Tutkinto-ohjelma:	Sähkö- ja automaatiotekniikka
Ammatillinen pääaine:	Automaatiotekniikka
Ohjaajat:	Yliopettaja Erkki Räsänen

Insinöörityön tarkoituksena oli perehtyä ympäristöolosuhteiden mittaamiseen ja tiedon pilvitietokantaan tallentamiseen MQTT-protokollaa ja Raspberry Pi -pienoistietokonetta käyttäen. Työssä käydään vaiheittain läpi koko dataputki aina mittauskytkennöistä pilvitietokantaan tallentamiseen ja visualisointiin saakka.

Mittaukseen tarvittavat komponentit kytkettiin Raspberry Pi -pienoistietokoneeseen, johon myös asennettiin MQTT-protokollan mukaiset asiakas- ja välittäjäohjelmistot. Anturien lukeminen tapahtui laitteistossa ajetulla Python-ohjelmalla, josta mittadata lähetettiin eteenpäin Node-RED-ohjelmalle MQTT-protokollan toiminnan ja mahdollisuuksien havainnollistamiseksi.

Insinöörityön lopputuloksena oli valmis Raspberry Pi -pohjainen kokonaisuus ympäristöolosuhteiden mittaamiseen ja MongoDB Atlas -pilvitietokantaan tallentamiseen. Kokonaisuus on helposti laajennettavissa ja voi toimia pohjana erilaisille automaatioratkaisuille. Työssä käydään läpi vaiheittain tarvittavien komponenttien ja ohjelmistojen asennukset ja käyttöönotot ja työtä on kirjoitettu niin, että sitä on helppo käyttää ohjeistuksena vastaavia mittalaitteistoja tehdessä.

Avainsanat: Raspberry Pi, MQTT, MongoDB, IoT, tietokanta

Abstract

Author:	Miika Sinervä
Title:	Collecting Measurement Data to a Cloud Database Using Raspberry Pi and the MQTT Protocol
Number of Pages:	37 pages + 2 appendices
Date:	30 April 2025
Degree:	Bachelor of Engineering
Degree Programme:	Electrical and Automation Engineering
Professional Major:	Automation Engineering
Supervisor:	Erkki Räsänen, Principal Lecturer

The purpose of this thesis was to explore the measurement of environmental conditions as well as how to store the data into cloud database using MQTT protocol and Raspberry Pi single-board computer. The thesis covers the entire data pipeline step by step, from connecting the sensors to storing the data into the cloud database and visualizing it.

The components required for the measurements were connected to the Raspberry Pi, which also had the MQTT protocol-based client and broker software installed on it. Reading the sensors was done using a Python program that was running on the hardware, and the measurements were sent forward to a Node-RED program to demonstrate the operation and possibilities of MQTT protocol.

The result of the thesis was a complete Raspberry Pi-based system for measuring environmental conditions and storing the data in MongoDB Atlas cloud database. The system is easily expandable and can work as a foundation for various automation solutions. The thesis includes the installation and setup of the required components and software step by step, and is written in a way that it can be used as a guide when creating similar measurement systems.

Keywords: Raspberry Pi, MQTT, MongoDB, IoT, database

Sisällys

Liite 2: InsertOne-noodin yhteysmäärittelyt

Lyhenteet

GPIO: *General-Purpose Input-Output.* Yleiskäyttöinen pinni, joka on ohjelmoitavissa joko sisääntuloksi tai ulostuloksi.

I2C: *Inter-Integrated Circuit.* Kaksisuuntainen ja sarjamuotoinen ohjaus- ja tiedonsiirtoväylä.

IoT: *Internet of Things.* Esineiden internet.

JSON: *JavaScript Object Notation.* Ohjelmointikielestä riippumaton avoimen standardin tiedostomuoto tiedonvälitykseen.

MQTT: *Message Queuing Telemetry Transport.* Avoimen lähdekoodin kevyt julkaisija-tilaaja-mallin tiedonsiirtoprotokolla.

NoSQL: *Not Only SQL.* Perinteisestä relaatiotietokannasta poikkeava tietokanta, jonka ei tarvitse noudattaa ennalta määrättyä tietorakennetta ja joka tukee joustavia tietomalleja.

SQL: *Structured Query Language.* Standardoitu tietokantojen hallintaan käytetty kyselykieli.

1 Johdanto

Automaatiojärjestelmien elinehtona on reaaliaikainen data ja sen hyödyntäminen automaattisten ohjauksien toteuttamisessa ennalta määrättyjen algoritmien mukaisesti. Käyttökelpoisen datan saamiseksi on suoritettava mittauksia, mittaustuloksien käsittelyä ja tulkitsemista. Suurella määrällä mittausdataa nähdään usein olevan arvoa prosessin ymmärtämisen ja tehostamisen kannalta, jonka vuoksi data täytyy siirtää ja säilöä tulevaa analysointi varten.

Tässä opinnäytetyössä tarkastellaan ympäristöolosuhteiden mittaamista ja pilvitietokantaan tallentamista Raspberry Pi -pienoistietokonetta käyttäen. Mitattavina suureina ovat lämpötila, ilmankosteus sekä valaistusvoimakkuus. Anturit kytketään Raspberry Pi:n GPIO-pinneihin ja mittauksien lukemiseen ja käsittelyyn käytetään Python-ohjelmointikieltä. Raspberry Pi:lle asennetaan paikallinen MQTT-välittäjä, jonka avulla mittausdataa siirretään Python-ohjelmasta Node-RED-ohjelmalle. Node-RED-ohjelman avulla tieto viedään edelleen MongoDB Atlas -pilvitietokantaan myöhempää käyttöä varten. Mittausdatan historiatietojen visualisointinäyttö toteutetaan Node-RED-ohjelmalla.

Työn tavoitteena on luoda olosuhteiden mittaamiseen soveltuva kustannustehokas anturointipaketti ja MQTT-protokollaa hyödyntävä valmis dataputki, jolla tieto saadaan välitettyä aina visualisointiin ja pilvitietokantaan saakka. Opinnäytetyössä käydään vaiheittain läpi ympäristöolosuhteiden mittaamisen koko dataprosessi.

2 Raspberry Pi -pienoistietokone

Raspberry Pi on yhden piirilevyn monikäyttöinen pienoistietokone, jota kehittää hyväntekeväisyysjärjestö Raspberry Pi Foundation. Järjestön tavoitteena on mahdollistaa tietotekniikan opiskelu mahdollisimman monille. Raspberry Pi:n kehitystyössä pyritäänkin pitämään tuotteen hinta matalana. Ensimmäinen versio Raspberry Pi:stä julkaistiin vuonna 2012. Raspberry Pi:n käyttäjät usein

käyttävät sitä ohjelmointitaitojensa kehittämiseen, elektroniikka- ja kotiautomaatioprojektien toteuttamiseen, reunalaskentaan ja jopa teollisten sovelluksien toteuttamiseen. [1.]

2.1 Raspberry Pi:n tekniset tiedot

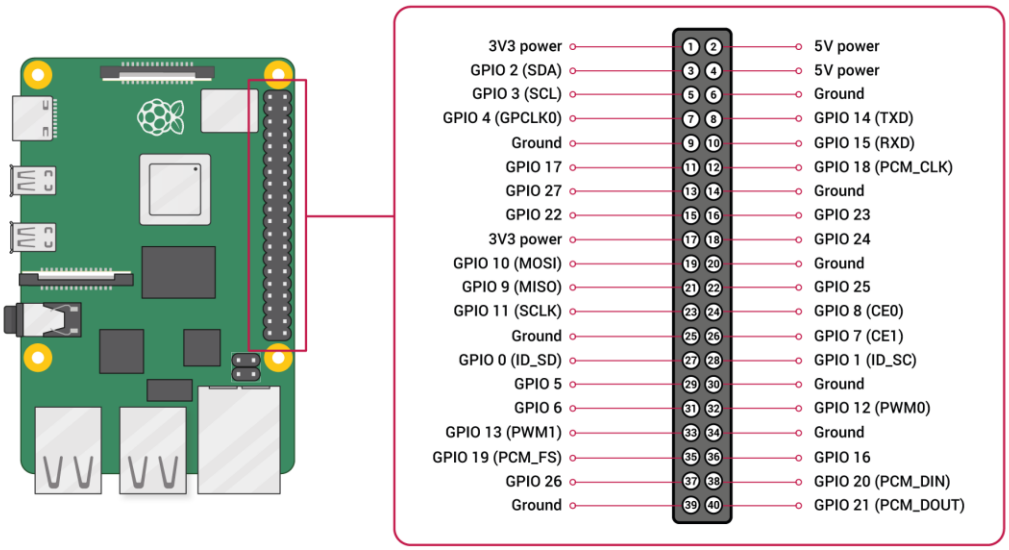
Kaikki tässä opinnäytetyössä käytetyt komponentit kytketään Raspberry Pi:hin, johon myös kaikki tarvittavat ohjelmistot asennetaan. Käytössä on Raspberry Pi 5, joka on Raspberry Pi tuoteperheen uusin versio. Tämä versio käyttää ARM-arkkitehtuuriin perustuvaa Broadcomin BCM2712 -prosessoria. Sitä on saatavilla 2, 4, 8 ja 16 gigatavun keskusmuisti vaihtoehdoilla [2]. Opinnäytetyössä käytetty malli on varustettu 8 gigatavun keskusmuistilla.

Virransyöttö vaihtoehtoina Raspberry Pi:lle ovat 25W (5V / 5A) syötettynä USB-C:llä tai GPIO-pinnin kautta syötetty viiden voltin tasajännite. Tuettuina yhteysvaihtoehtoina ovat Bluetooth, 2,4/5GHz Wi-Fi sekä Gigabit Ethernet RJ45-liittimellä. Tietojen tallennusta varten pienoistietokoneesta löytyy microSD-kortinlukija. Näytöille, hiirelle, näppäimistölle ja muille mahdollisille lisälaitteille löytyy seuraavia liitäntöjä:

- 2 kpl mikro-HDMI-liittimiä
- 2 kpl USB 2.0 -liittimiä
- 2 kpl USB 3.0 -liittimiä
- 1 kpl PCIe-portteja
- 40 kpl GPIO-pinnejä
- 2 kpl 22-pinnisiä CSI/DSI-portteja kameroille.

GPIO on monikäyttöinen pinni, joka on ohjelmoitavissa joko tuloksi tai lähdöksi, jonka lisäksi pinneissä voi olla muita erillisiä toimintoja. GPIO-pinnejä on

Raspberry Pi 5:ssä 40 kappaletta (kuva 1). [2.]



Kuva 1. GPIO-pinnien toiminnallisuudet. [2.]

Kuvasta 1 nähdään GPIO-pinnien keskinäinen järjestys piirilevyllä ja eri pinnien sisältämiä toimintoja ja käyttötarkoituksia.

2.2 Raspberry Pi:n käyttöjärjestelmä

Raspberry Pi on suunniteltu toimimaan parhaiten Linux-pohjaisilla käyttöjärjestelmillä. Käyttäjärjestelmä vaihtoehtoja on useita, mutta yksi suosituimmista lienee yleiskäyttöinen Raspberry Pi Foundationin kehittämä Debian-pohjainen Raspberry Pi OS. Raspberry Pi OS tarjoaa esiasennettuna useita tarpeellisia perusohjelmistoja, kuten tekstinkäsittelyohjelma, verkkoselain ja kehitysympäristö Python-ohjelmointiin. Käyttöjärjestelmä sisältää tarvittavat rajapinnat, joiden avulla GPIO-pinnejen käyttö on yksinkertaista. [3.] Raspberry Pi:n eri käyttötarkoituksia varten voi olla perusteltua valita eri käyttöjärjestelmä, mutta Raspberry Pi OS -käyttöjärjestelmän suosion vuoksi hyvin monet Internetistä löytyvät IoT- ja elektroniikkaprojektien ohjeet on tehty juuri kyseiselle käyttöjärjestelmälle, joten se on monessa tapauksessa hyvä oletusvalinta.

Komentorivin käyttö tietokoneen käskyttämisessä on suosittu ja tehokas tapa tehdä asioita Linux-ympäristössä. Graafiseen käyttöliittymään verrattuna suorat komentorivillä suoritetut komennot ovat usein nopeampi ja tehokkaampi tapa navigoida järjestelmässä tai suorittaa haluttuja toimintoja ja joitain toimintoja ei pysty graafisen käyttöliittymän kautta suorittamaan lainkaan. Tyypillisiä komentorivin kautta suoritettuja toimintoja ovat mm. kansioiden välillä navigointi, tiedostojen luominen, siirtäminen ja poistaminen, SSH-etäyhteyden luominen, ohjelmistoskriptien ajaminen sekä ohjelmistojen ja päivityksien asentaminen. [4.]

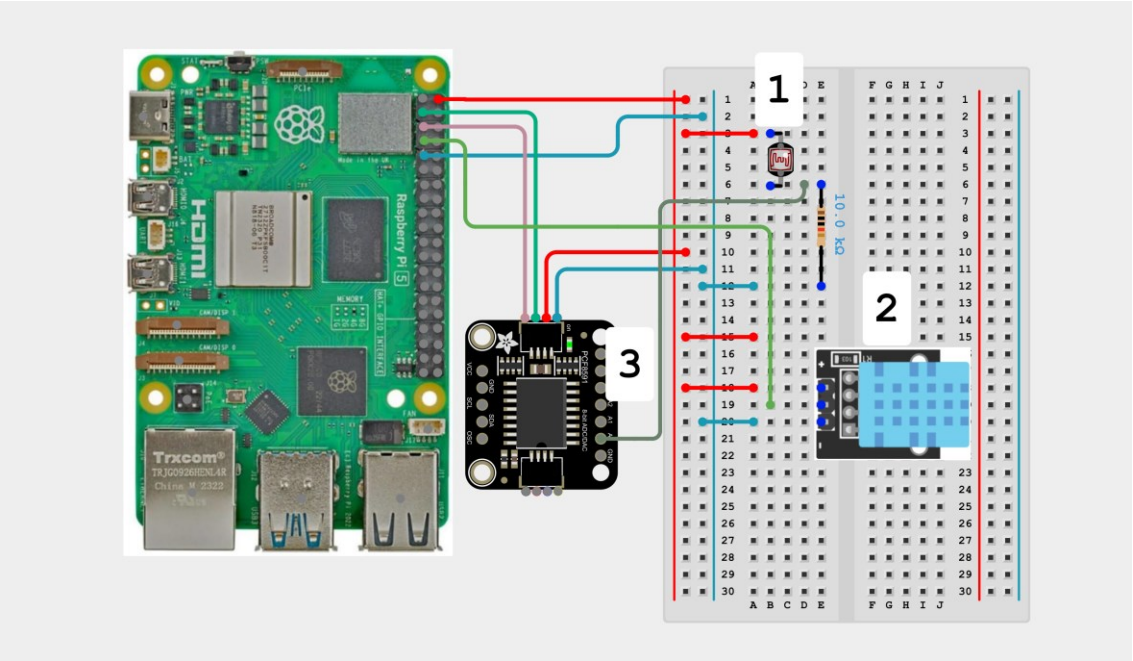
3 Ympäristöolosuhteiden mittaaminen

3.1 Elektroniset komponentit ja kytkennät

Tässä luvussa käsitellään ympäristöolosuhteiden mittaamiseen tarvittavia komponentteja, toimintaperiaatteita ja komponenttien ja Raspberry Pi -pienoistietokoneen välisiä sähköisiä kytkentöjä. Luvussa käsitellään myös analogiasignaalin muuntamista digitaaliseksi ja muunninmoduulin ja Raspberry Pi:n välistä sarjaliikennekommunikaatiota.

Ympäristön valaistuksen voimakkuuden mittaaminen

Ympäristön valaistuksen voimakkuuden mittaaminen toteutetaan käyttämällä valovastusta. Valovastus on puolijohdekomponentti, jonka resistanssi muuttuu siihen osuvan valaistuksen voimakkuuden mukaan. Valon osuessa valovastukseen valon fotonien energia vapauttaa puolijohteen elektroneja, jolloin valovastuksen resistanssi pienenee. [5.] Valovastus ja kiinteä vastus voidaan kytkeä sarjaan jännitelähteen ja maapisteen välille. Näin on mahdollista toteuttaa jännitteenjakokytkentä. Valon osuessa valovastukseen sen resistanssi pienenee, jolloin jälkimmäisen kiinteän vastuksen yli oleva jännite kasvaa. Mittaamalla jännitettä vastuksien välistä (Kuva 2) pystytään mittaamaan valaistuksen voimakkuutta, jolloin suurempi jännitearvo tästä pisteestä mitattuna tarkoittaa valoisampia olosuhteita. [6.]



Kuva 2. Raspberryn ja mittakomponenttien kytkentäkaavio. Numerointien selitteet: 1. valovastus 2. DHT11-moduuli sisäänrakennetulla 10kΩ ylösvetovastuksella 3. AD/DA-muunninmoduuli PCF8591.

Valovastuksen tuottaman analogisen signaalin käsittelyyn ja välittämiseen käytetään 8-bittistä AD/DA-muunninmoduulia, jonka malli on PCF8591. Muunninmoduuli ottaa sisään analogisen 0–5 V mittasignaalin ja muuntaa signaalin diskreetiksi digitaaliseksi viestiksi. Muuntimen 8-bittisyyden vuoksi analogia-digitaalimuunnoksen tuottama digitaalisignaali voi saada $8^2 = 256$ eri arvoa, joka tarkoittaa sitä, että digitaaliviesti on kokonaisluku väliltä 0–255. Valovastukselta mitattu ja muunnettu lukuarvo on tunnukseton suure ja jännitteenjaon perusteella, mitä suurempi lukuarvo on, sitä voimakkaampi on ympäristön valaistus.

Lämpötilan ja ilmankosteuden mittaaminen

Lämpötilan ja ilmankosteuden mittaukseen tässä työssä käytetään digitaalista DHT11-yhdistelmäanturia, joka mittaa ilmankosteutta ja lämpötilaa. DHT11-yhdistelmäanturi käyttää ilmankosteuden mittaamiseen kapasitiivista kosteusanturia, joka perustuu kahden elektrodin välisen resistanssin mittaukseen. Korkeampi ilmankosteus johtaa elektrodien välisen resistanssin pienenemiseen, jonka perusteella suhteellinen kosteus saadaan laskettua. Lämpötilan mittaukseen

DHT11 puolestaan käyttää NTC-termistoria (Negative Temperature Coefficient). NTC-termistorilla lämpötilan ja resistanssin välinen suhde on käänteisesti verrannollinen, jolloin lämpötilan noustessa termistorin resistanssi laskee. Ilmankosteudelle ilmoitettu mitta-alue DHT11-anturilla on 20–90 % tarkkuuden ollessa ±5 % ja mitatun lämpötilan mitta-alueeksi on ilmoitettu 0–50 °C mittaustarkkuuden ollessa ±2 %. DHT11-moduuli sisältää myös 8-bittisen mikrokontrollerin, joka muuttaa lämpötilan ja ilmankosteuden analogiset arvot digitaaliseen sarjaliikennemuotoon. [7; 8, s. 1.]

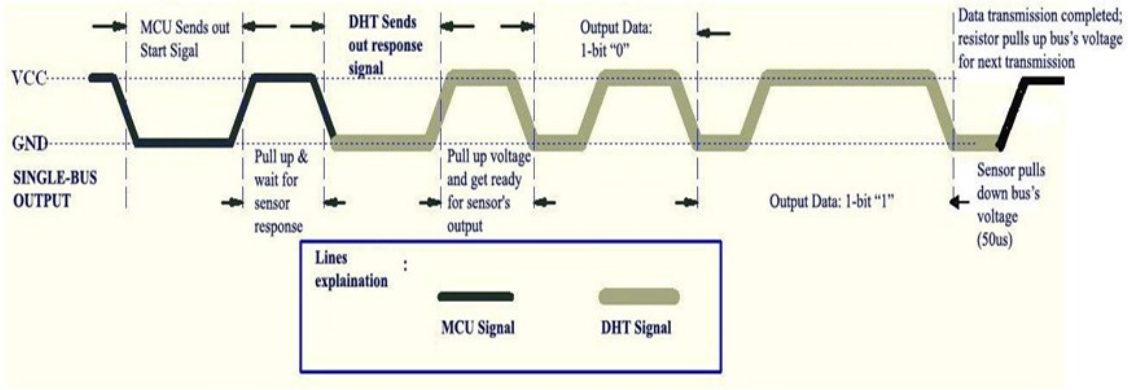
DHT11-anturia on saatavissa 3-pinnisenä, jolloin pinnien järjestys on käyttöjännite, datapinni ja maadoituspinni tai 4-pinnisenä, jolloin datapinnin ja maadoituspinnin välissä on ylimääräinen käyttämätön pinni. Valittavana on joko erillinen DHT11-anturi tai valmis DHT11-moduuli, jossa on sisäänrakennettuna datapinnin yhteyteen ylösvetovastus sekä suodatinkondensaattori. [7.] Datapinni on se pinni, jonka kautta mittausdataa siirretään. Mikäli valmista moduulia ei käytetä niin valmistaja suosittelee käyttämään 5–10 kilo-ohmin ylösvetovastusta datapinnin ja käyttöjännitteen välissä, jotta viestisignaali ei jää ”kellumaan” epämääräiseen tilaan arvojen välille. Käyttöjännitteen sekä maapisteen välissä suositellaan käyttämään 100 nanofaradin suodatinkondensaattoria suodattamaan jännitevaihteluita. [8, s. 5.]

3.2 Mittausanturien ja Raspberry Pi:n välinen kommunikaatio

Tässä luvussa käsitellään PCF8591-muunninmoduulin digitaaliseksi muuntaman signaalin ja DHT11-anturin mittaustuloksien välittämistä Raspberry Pi -pienoistietokoneelle. PCF8591-muunninmoduulin ja Raspberry Pi -pienoistietokoneen välisessä kommunikaatiossa käytetään I2C-sarjaliikenneprotokollaa (Inter-Integrated Circuit). Väylän tarvitsemat kaksi signaalijohdinta liitetään Raspberry sekä muunninmoduulin SDA- ja SCL-pinneihin, joista toinen on signaali kuljettaa kellopulssia ja toista käytetään binääridatan siirtämiseen. I2C-väylän kytketyt laitteet ovat aina joko käskyttäviä isäntälaitteita (master) tai käskytettäviä orjalaitteita (slave). Isäntälaite vastaa kommunikaation aloittamisesta ja kellolinjan pulssituksesta, kun taas orjalaitteen käskytettävän laitteen tehtävänä on vastata

isäntälaitteen pyyntöihin. PCF8591:ltä lukemiseen I2C-väylällä tarvitaan laitteen 7-bittinen väyläosoite ja luettavan laitteen ensimmäisen analogiatulon sisäisen rekisterin osoite. [9.]

DHT11-yhdistelmäanturi käyttää viestintään yhden johtimen kaksisuuntaista sarjaliikenneprotokollaa. Tiedonsiirron alussa, kun mikrokontrolleri tai tässä tapauksessa Raspberry Pi haluaa lukea mittadataa, sen on aloitettava tiedonsiirto. Raspberry Pi lähettää vähintään 18 mikrosekunnin nollapulssin DHT11-anturille (Kuva 3), jonka jälkeen Raspberry Pi jää odottamaan DHT11-anturin vastausta. Anturi vastaa Raspberry Pi:n lähettämään nollapulssiin 80 mikrosekunnin nollapulssilla, jonka jälkeen odottaa 80 mikrosekuntia, kunnes alkaa lähettämään 40 bitin mittadataviestiään. Yhden välitetyn bitin lähettämiseen kuuluu 50 mikrosekunnin odotusaika, jolloin viestijohtimen jännite on matalalla tasolla ja aika, jolloin jännitetaso on korkealla.



Kuva 3. DHT11-anturin käyttämä sarjaliikenneprotokolla pääpiirteisesti visualisoituna [8, s. 6].

Korkean jännitetason kesto määrittelee bitin arvon, 26–28 mikrosekunnin aika vastaa arvoa 0 kun taas 70 mikrosekunnin aika vastaa arvoa 1. Data on jaettu viiteen 8 bitin osaan: kosteusanturin kokonaislukuosa, kosteusanturin desimaalilukuosa, lämpötila-anturin kokonaislukuosa, lämpötila-anturin desimaalilukuosa ja viimeisenä tarkistussumma. [8, s. 5–8.]

3.3 Mittadatan lukeminen ja käsittely Pythonilla

Tässä luvussa käsitellään mittausdatan lukemista antureilta käyttäen Python-ohjelmointikieltä ja sitä, kuinka mittadata tulee käsitellä ennen kuin se on valmis lähetettäväksi eteenpäin seuraavalle ohjelmalle tai pilvitietokantaan. Luvussa käydään läpi tarvittavat Python-kirjastot ja mittadatan syntaksin yhtenäistäminen JSON-objektiksi.

Valovastuksen mittadatan lukeminen

Ympäristöolosuhteiden mittadatan lukeminen antureilta ja mittadatan käsittely toteutetaan Python-ohjelmointikielellä. Mittauksiin käytetyn valovastuksen ja AD-muunninmoduulin tuottaman digitaaliviestin lukemiseen käytetään Pythonin smbus-kirjaston valmiita funktioita. Smbus-kirjasto, saadaan asennettua pip-paketinhallintajärjestelmää käyttäen kirjoittamalla käyttöjärjestelmän komentoriville komento ”pip install smbus”, jonka jälkeen kirjasto saadaan Python-ohjelman sisällä käyttöön tuomalla se ohjelmaan import-komennolla. I2C-väylän lukemista varten ohjelmassa määritetään PCF8591-laitteen I2C-osoite, joka on heksadesimaalilukuna 0x48 ja lisäksi ensimmäiseen analogiatuloon kohdistuva lukukäsky 0x40 (Kuva 4). Väylä alustetaan bus-muuttujaan. [10, s. 67.]

```
1   import time
2   import smbus
3
4   address = 0x48
5   A0 = 0x40
6   bus = smbus.SMBus(1)
7
8   while True:
9       bus.write_byte(address,A0)
10      brightness = bus.read_byte(address)
11      time.sleep(2)
```

Kuva 4. PCF8591-muunninmoduulilta lukeminen ja arvon tallentaminen.

I2V-väylän yhteysmäärittelyjen jälkeen Python-ohjelma ohjelmoidaan lähettämään muunninmoduulin analogiatuloon kohdistuva lukupyyntö laitteelle, jonka jälkeen lukupyynnön lopputulos tallennetaan muuttujaan. Tämän jälkeen odotetaan 2 sekuntia, kunnes sama ohjelmakierros toistetaan uudelleen.

DHT11-anturin mittadatan lukeminen

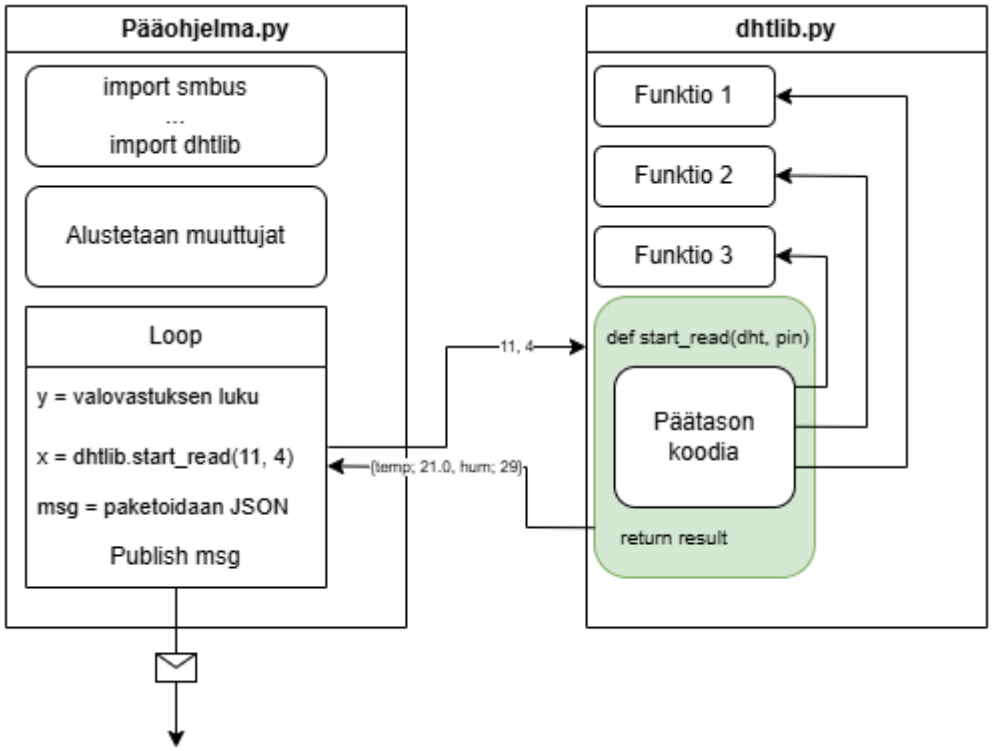
DHT-viestiä ymmärtääkseen Python-ohjelma tarvitsee erillisen ohjelmakirjaston. Raspberryille suunniteltuja DHT-kirjastoja on useita vaihtoehtoja, kuten Adafruitin tai Circuitpythonin kirjastot. DHT11-anturin lukemiseen soveltuvia vaihtoehtoja kartoittaessa todettiin, että yleisimpiä ohjeissa mainittuja kirjastoja ei ole vielä päivitetty toimivaksi Raspberry 5:llä, jonka vuoksi täysin valmista DHT11-anturin lukemiseen soveltuvaa kirjastoratkaisua ei ole käytettävissä.

Ohjelmakirjaston pohjaksi otetaan Zoltan Szarvasin [11] luoma ratkaisu DHT-anturien lukemiselle. Ratkaisu sisältää Python-ohjelmointikielen oliomäärittelyjä, funktioita ja päätason koodia. Pohjaksi otettu ratkaisu ei ole suoraan valmis käytettäväksi kirjastona sen sisältämän päätason koodin takia. Ratkaisua voidaan ajaa pääohjelmamme kautta Pythonin subprocess-moduulilla, mutta tämä avaa ohjelman täysin omana erillisenä prosessinaan taustalle. Tämä tarkoittaa, että prosessille käynnistyy oma erillinen Python-tulkki, joka ei ole resurssitehokasta, koska uusi tulkki ei jaa muistia tai muuttujia alkuperäisen pääohjelman kanssa.

Päätason koodin vuoksi pohjana käytettävää ratkaisua ei voida suoraan kutsua kirjastona, koska jo kirjaston tuominen pääohjelmaan saattaa suorittaa päätasolla sijaitsevan koodin. Pohjana käytettyyn ratkaisuun tehdään pieniä muutoksia, jotta sitä voidaan käyttää valmiina kirjastona Python-pääohjelman sisällä ilman, että varsinaista kirjaston sisällä olevaa koodia pitää tallentaa pääohjelmaan sisälle. Käytännössä jotta Zoltan Szarvasin luomaa ratkaisua voidaan käyttää ohjelmakirjastona riittää, että ratkaisussa päätasolla oleva koodipätkän paketoidaan omaksi funktiokseen, jota nimetään nyt nimellä ”start_read”.

Uusi start_read-funktio tarvitsee sisään otettavina parametreinaan DHT-anturin mallinumeron eli tässä tapauksessa ’’11’’, sekä pinninumeron, johon anturi on

kytketty Raspberry Pi:ssä. Tämän jälkeen ohjelma toimii kuten ennekin kutsuen muita sisäisiä funktioitaan, kunnes lopulta palauttaa DTH-anturin mittaustulokset varsinaiselle Python-pääohjelmalle (Kuva 5).



Kuva 5. Pääohjelman ja DHT-anturikirjaston vuokaavioesitys.

Nimetään uusi kirjastotiedosto vapaavalintaisella nimellä ”dhtlib.py”. Kirjastotiedosto tallennetaan samaan kansioon pääohjelman kanssa, jolloin ohjelmaa voidaan käyttää kirjastona tuomalla se pääohjelmaan komennolla ”import dhtlib” ja kutsumalla start_read-funktiota.

JSON-tietorakenne

JSON (JavaScript Object Notation) on JavaScript-ohjelmointikielestä lähtöisin oleva, mutta nykyään ohjelmointikielestä riippumaton standardoitu tiedostomuoto. JSON-formaatti käyttää avain-arvo-pareihin perustavaa syntaksia datalle, joka tekee siitä helppolukuista ja käyttökelpoista, koska JSON-formaatti on hyvin tuettu eri ympäristöjen työkaluilla. [12.]

JSON-dokumentissa mahdollisia tietotyyppejä ovat objekti, lista, merkkijono, luku, boolean ja null. Rakenteeltaan JSON-objekti vastaa Python ohjelmointikielen sanakirja-muuttujatyyppiä. Pythonissa JSON-objektien käsittelyyn käytetään json-kirjastoa, joka tuodaan ohjelmaan komennolla ”import json”. Pythonin json-kirjaston avulla Python muuttujatyyppejä voidaan suoraan muuntaa vastaaviksi JSON-tyypeiksi. JSON-formaatin syntaksissa avain-arvo-parit on järjestetty aaltosulkeiden väliin pilkuilla erotettuna (Kuva 6). [12.]

Uusi luotu Pythonin dhtlib-kirjasto on muokattu palauttamaan pääohjelmalle lämpötila-arvo ja ilmankosteus sanakirja-tietotyypissä. Tämän jälkeen pääohjelmassa sanakirjaan lisätään avain-arvo-pari valaistukselle, jonka jälkeen sanakirja muutetaan sen eteenpäin lähettämistä varten sanakirjasta varsinaiseksi JSON-objektiksi kirjaston dump-funktiolla.

```
22 while True:
23     bus.write_byte(address,A0)
24     value = bus.read_byte(address)
25     msg = dhtlib.start_read(dhtnum, dhtpin)
26     msg["brightness"] = value
27     payload = json.dumps(msg)
28     print(f'Viesti:', payload)
29     #client.publish("sensors", payload = json.dumps(output))
30     time.sleep(2)
31
```

Shell

```
Viesti: {"temperature": 23.4, "humidity": 13.0, "brightness": 128}
Viesti: {"temperature": 23.4, "humidity": 13.0, "brightness": 178}
Viesti: {"temperature": 23.5, "humidity": 13.0, "brightness": 179}
Viesti: {"temperature": 23.6, "humidity": 12.0, "brightness": 179}
```

Kuva 6. Tietojen koonti JSON-objektiksi ja objektin tulostus syntaksin hahmottamiseksi.

Kuvan 6 ikkunan alareunassa on tulostettuna JSON-objektin sisältö ja rakenne. Mittaustuloksiin viitataan käyttämällä mittaustulosta vastaavaa avainta; temperature, humidity tai brightness. Nyt Python-ohjelmassa on kaikki, mikä mittatiedon lukemisen, käsittelyn ja paketoinnin kannalta on tarpeellista. Mittaviesti on nyt valmis lähettäväksi eteenpäin MQTT-välittäjän kautta Node-RED-ohjelmalle.

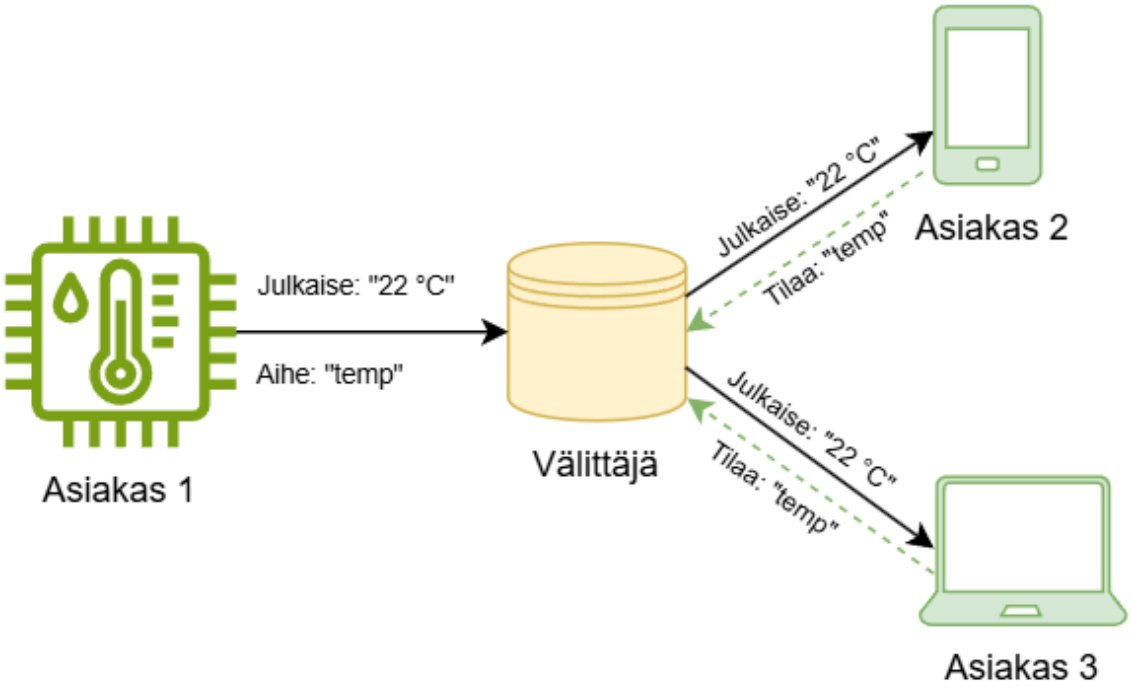
4 MQTT-protokolla

Tässä luvussa käsitellään MQTT-protokollan toimintaperiaatteita ja hyötyjä IoT-sovelluksissa. Luvussa käydään läpi MQTT-protokollan mukaisten osapuolten tehtäviä sekä joitain protokollan ominaisuuksista. Lopuksi käydään läpi MQTT-välittäjän käyttöönotto ja viestien julkaiseminen Python-ohjelmalla.

4.1 Julkaisija-tilaaja-arkkitehtuuri MQTT-protokollassa

MQTT-protokolla on kevyt julkaisija-tilaaja-mallin viestintäprotokolla. MQTT kehitettiin vuonna 1999 kun öljy- ja kaasuteollisuudessa syntyi tarve saada öljylinjojen mittadataa siirrettyä tehokkaasti käyttämättä suurta määrää tietoliikennekapasiteettia. Samalla periaatteella MQTT-protokollaa hyödynnetään tänäkin päivänä ja erityisesti se on suosiossa IoT- ja IIoT-sovelluksissa, koneiden välisessä kommunikaatiossa sekä älysensorien ja puettavien älylaitteiden kommunikaatiossa. [13.]

Asiakkaiden välissä olevan välittäjän takia tilaajat ja julkaisijat ovat erotettuna toisistaan (Kuva 7). Erotuksen takia tilaajien ja julkaisijoiden ei tarvitse olla tietoisia toistensa sijainnista verkossa, osapuolten ei tarvitse olla yhteydessä verkkoon samanaikaisesti, ja osapuolet voivat lähettää viestejä ilman tarvetta synkronisoida toimintaa. Sanotaan siis, että osapuolet ovat tilallisesti, ajallisesti ja synkronisesti erotettuina. Tämä mahdollistaa MQTT-verkolle hyvän skaalautuvuuden, koska uusien laitteiden tarvitsee luoda yhteys ainoastaan välittäjään sen sijaan, että kaikkien laitteiden tulisi olla suorassa yhteydessä toisiinsa. [13.]



Kuva 7. MQTT-protokollan mukainen julkaisija-tilaa-arkkitehtuuri.

Kuvassa 7 on hahmoteltuna tilanne, jossa yksi julkaisija-asiakas julkaisee lämpötilatietoa aiheotsikolla ”temp” ja kaksi asiakasta on tilannut kyseisen aiheen. Välittäjä vastaanottaa aiheeseen julkaistut viestit ja välittää viestit eteenpäin kaikille tilaajille aina silloin kun uusi viesti julkaistaan.

4.2 Välittäjä ja asiakas MQTT-protokollassa

Välittäjän rooli

Tässä luvussa käsillään välittäjän roolia ja tehtäviä MQTT-protokollassa. Välittäjä eli broker toimii MQTT-protokollan keskiössä yhdistämässä asiakkaita toisiinsa viestin välittämiseksi. Sen pääasiallinen vastuu on välittää verkkoon julkaistuja viestejä julkaisijoilta tilaajille. Välittäjä siis vastaanottaa verkkoon lähetetyt viestit, suodattaa aiheen mukaan ja lähettää edelleen oikeille aiheiden tilaajille.

Välittäjäohjelmistoja on useita erilaisia ja sopivan välittäjäohjelmiston valinnassa on huomioitava välittäjän eri tehtäviä. Välittäjän tehtäviä ja ominaisuuksia ovat muun muassa:

- Asiakasyhteyksien hallinta: Välittäjällä on parhaimmillaan kyky hallita miljoonia yhtäaikaisia asiakasyhteyksiä hyödyntäen kuormantasaus ominaisuuksia.

- Lähetettyjen viestien suodatus ja reititys: Viestit suodatetaan tilausaiheen otsikon perusteella ja määritetään, mitkä asiakkaat saavat viestin.

- Istuntojen hallinta: Yhdistettyjen asiakkaiden istuntotietojen säilyttäminen, kuten tilatut aiheet tai viestien säilöminen, mikäli yhteysasetuksissa on määritetty ohi menneet viestit säilytettäviksi.

- Asiakkaiden autentikointi ja valtuutus: Asiakkaiden tunnistaminen ja valtuutus asiakkaan toimittamien tunnistetietojen perusteella. Laajat mahdollisuudet eri tietoturvaominaisuuksiin, kuten viestien salaukseen tiedonsiirron aikana ja erilaiset käyttöoikeuslistat.

- Verkon laajennettavuus ja monitorointi: MQTT-välittäjäohjelmiston on oltava helposti skaalautuva, jotta se mahdollistaa suurien viesti- ja asiakasmäärien hallinnan. Välittäjäohjelmistossa tarjolla voi olla valvontaan liittyviä ominaisuuksia, kuten reaaliaikaisen analytiikan työkaluja.

- Välittäjän klusterointi: Mahdollisuus ajaa useampaa instanssia välittäjästä rinnakkain, mahdollistaen suuremman prosessointikapasiteetin asiakkaiden ja viestien käsittelyyn. Käytännössä tämä tarkoittaa, että välittäjäohjelmistosta ajetaan useaa kopiota rinnakkain ja nämä toimivat yhdessä ja jakavat kuorman. [14.]

MQTT-välittäjäohjelmistoja on saatavilla useita erilaisia, kuten AWS IoT Core MQTT, Mosquitto, Mosca/Aedes, HiveMQ, VerneMQ ja useita muita. Välittäjä voi toimia pilvipalvelussa palveluntarjoajan ylläpitämänä, jolloin palveluntarjoaja

huolehtii turvaominaisuuksista ja infrastruktuurista rahallista maksua vastaan. Toinen vaihtoehtoinen tapa on käyttää MQTT-välityspalvelinta paikallisesti, jolloin välitysohjelmisto asennetaan omalle laitteelle ja huolehditaan turvaominaisuuksista ja konfiguroinnista itse. [15.]

Asiakkaan rooli

MQTT-protokolla on rakennettu toimimaan TCP/IP-protokollapinon päällä. Asiakas eli client voi siis olla mikä tahansa laite tai ohjelmisto, joka ymmärtää TCP/IP-protokollaa ja käyttää MQTT-protokollaa lähettääkseen tai vastaanottaakseen viestejä välittäjältä. Asiakkaita on kahdenlaisia, julkaisijoita, jotka julkaisevat viestejä sekä tilaajia, jotka vastaanottavat haluamaansa aiheeseen liittyviä viestejä. Protokollassa on mahdollista, että asiakas on samaan aikaan sekä julkaisija, että tilaaja. [16.]

Tämän työn kannalta asiakkaiksi voidaan mieltää siis sekä Python-ohjelma, joka vastaa mittaustuloksien lukemisesta, paketoinnista ja lähettämisestä sekä Node-RED-ohjelma, joka tilaa aiheet ja vastaanottaa viestit.

4.3 MQTT-protokollan ominaisuuksia

4.3.1 Aihe ja villikortit

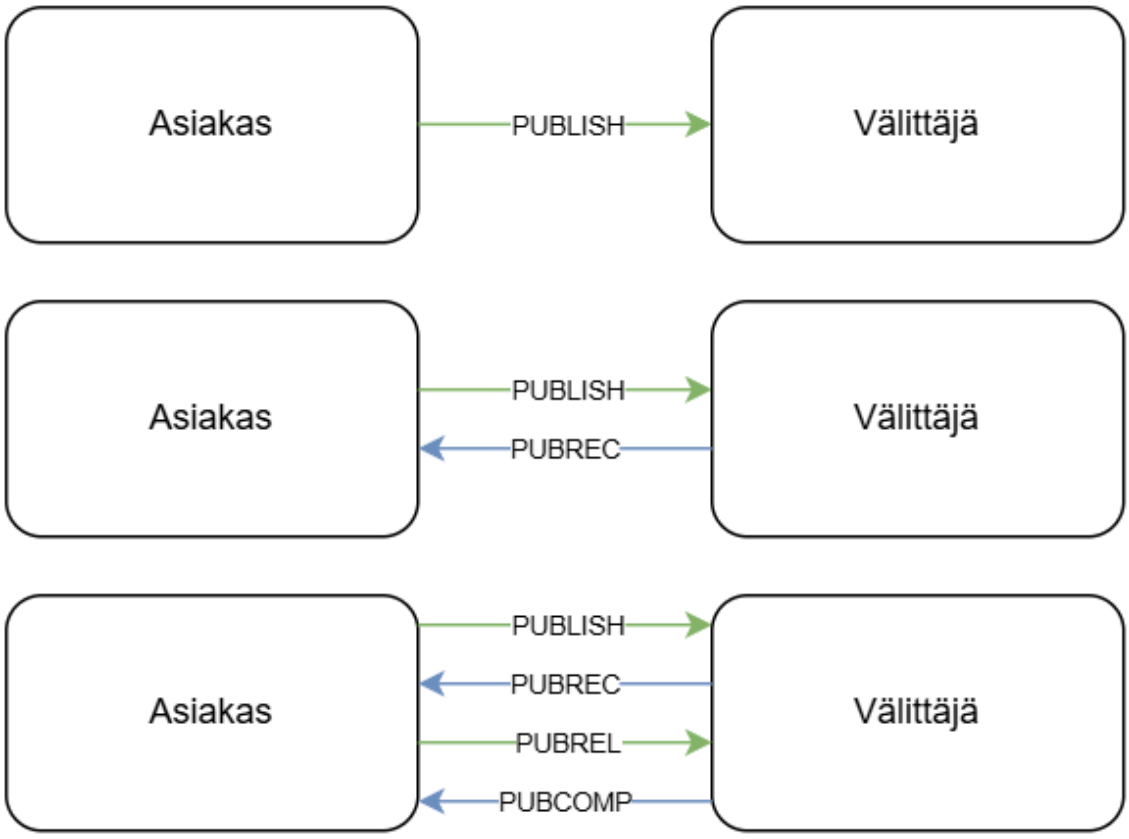
Aiheet toimivat hierarkkisesti ja ovat UTF-8-standardin mukaisia merkkijonomuuttujia. Aiheet ovat yksi- tai monitasoisia ja eri tasot ovat eroteltu /-merkillä, jolloin aihepolku muistuttaa tyypillistä kansiorakennetta. Aiheita voidaan tilata suoraan täsmällistä aihetunnistetta käyttäen tai niin sanottuja villikortteja hyödyntämällä. [17.] Kotiautomaatiojärjestelmässä aiheita voisi esimerkiksi olla koti/olohuone/lampotila, koti/keittio/lampotila tai koti/olohuone/valaistus.

Villikortilla tarkoitetaan symbolia, joka voi vastata mitä tahansa arvoa. Monitasoinen villikortti merkitään #-merkillä ja sen tulee sijaita aihepolun lopussa ja sillä voidaan korvata useampi taso aiheen hierarkiasta. Yksitasoinen villikortti puolestaan merkitään +-merkillä ja sillä voidaan korvata yksi taso aiheen

hierarkiassa. Malliaiheita käyttäen siis ”koti/olohuone/#” vastaanottaisi viestit kaikista aiheista, jotka alkavat ”koti/olohuone/”, eli aiheet 1, ja 3 kun taas ”koti/+/lampotila” vastaanottaa viestit aiheista 1 ja 2. [17.]

4.3.2 Viestien toimituksen laatutasot

MQTT-protokollassa on lähetettyjen viestien välitykseen eri laatutasoja (Quality of Services), jotka määrittävät sen, kuinka vahvasti viestin välittymistä vastaanottajalle valvotaan ja käytännössä sen, kuinka monesti yksi ja sama viesti voi päätyä vastaanottajalle. Laatutasoja on kolme (Kuva 8), joilla kaikilla on oma toimintatapa ja vaatimukset. Laatutasoa valittaessa on huomioitava, että laatutaso määritetään erillisesti julkaisijalta välittäjälle ja välittäjältä tilaajalle. Lähetetyllä viestipaketilla voi siis olla yksi tai useampi eri laatutaso matkalla lähettäjältä tilaajalle. [18.]



Kuva 8. MQTT-protokollan mukaiset viestien lähetyksen laatutasot 0–2.

Matalimmalla laatutasolla eli tasolla 0 yksittäinen viesti voi saapua vastaanottajalle korkeintaan yhden kerran. MQTT-julkaisija lähettää viestin yhden kerran, eikä viestiä kuitata vastaanotetuksi. Tämä laatutaso on verkkoa vähiten kuormittava, koska edestakaisia kuittausviestejä ei lähetetä eikä kadonneita paketteja lähetetä uudestaan. Tätä laatutasoa käytettäessä ei voida tietää menikö viesti ikinä perille vai katosiko paketti matkalla. [18.]

Keskimmäisellä MQTT-protokollan viestien välityksen laatutasolla eli tasolla 1 yksi viesti välitetään vastaanottajalle vähintään kerran mutta on mahdollista, että viesti päätyy perille useammin kuin kerran. Lähettäjä lähettää viestin PUBLISH-paketissa ja jää odottamaan PUBACK-pakettia kuittaukseksi siitä, että viesti on saapunut perille. Lähettäjä säilöö viestiä ja lähettää sen uudelleen, mikäli vastaanottajan PUBACK-paketti ei saavu prokollan määrittämässä ajassa. Sama viesti saattaa saapua vastaanottajalle useita kertoja, jos PUBACK-paketti katoaa matkalla. [18.]

Korkeimmalla laatutasolla eli tasolla 2 varmistetaan, että yksittäinen viesti saapuu vastaanottajalle täsmälleen yhden kerran. Tämän varmistamiseen käytetään nelivaiheista kättelyä lähettäjän ja vastaanottajan välillä. Lähettäjä lähettää viestin PUBLISH-paketissa ja säilöö sen.

- Vastaanottaja kuittaa paketin saapuneeksi nyt PUBREC-paketilla.

- PUBREC-paketti saapuu alkuperäiselle lähettäjälle. Alkuperäisen viestin säilöminen lopetetaan ja viesti tuhotaan. Kuitataan PUBREC vastaanotetuksi PUBREL-paketilla.

- PUBREL-paketti saapuu vastaanottajalle, jolloin kaikki siirtoon liittyvät tilat voidaan unohtaa ja lähettää PUBCOMP-paketti ensimmäiselle osapuolelle, joka PUBCOMP-paketin vastaanottaessaan myös unohtaa kaikki siirtoon liittyneet tilat. Nyt yhden viestin välittämiseen vaadittu kommunikointiprosessi on päättynyt.

Korkein laatutaso on kaikista luotettavin, silloin kun viesti on tärkeää saada perille, mutta viestin välittyminen useammin kuin kerran aiheuttaa haittaa. Tätä laatutasoa käytettäessä lähetettyjä paketteja tulee kaikista laatutasoista eniten, jonka vuoksi verkon kuormitus on suurin. [18.]

4.3.3 Pysyvä istunto ja tilapäinen istunto

Tässä luvussa käsitellään MQTT-protokollan mukaisia eri istuntotyyppejä. Asiakkaan yhdistäessä välittäjään on mahdollista määrittää istunto pysyväksi tai tilapäiseksi. Yhteyttä muodostaessa tilapäinen yhteys alkaa aina tyhjästä, kun pysyvää istuntoa käytettäessä välittäjä säilyttää asiakkaan tilaamat aiheet ja muita kommunikaatioon liittyviä tietoja. Pysyvän istunnon käyttäminen keventää yhteyttä ottavan asiakkaan työkuormaa, kun määrittelyjä ei tarvitse, jokaisella kerralla tehdä uudelleen. Tämä voi olla tarpeen silloin, kun asiakaslaitteen prosessointikyky on matala tai kun asiakaslaitteelle halutaan saada myös ne viestit perille, jotka lähetettiin silloin kun yhteyttä ei ollut. [19.]

Säilytettäviä tietoja pysyvää istuntoa käytettäessä ovat:

- Istunnon olemassaolo itsessään yksilöitynä niin, että uudelleen yhdistäessä pystytään jatkamaan oikeaa istuntoa oikeasta tilasta.

- Kaikki ne aiheet, jotka asiakas on tilannut.

- Laatutasoilla 1 ja 2 lähetetyt viestit, joiden välittäminen on jäänyt kesken ja se tila, johon kommunikaatioprosessissa jäätiin.

- Välittäjä säilöö kaikki laatutasoilla 1 ja 2 lähetetyt viestit, jotka lähetettiin sillä välin, kun asiakas on ollut offline-tilassa, jotta viestit voidaan välittää, kun yhteys jälleen muodostetaan. [19.]

Asiakkaan ja välittäjän välistä yhteyttä luodessa pysyvä istunto asetetaan päälle tai pois päältä määrittämällä cleanSession-lippumuuttujalle arvo 0 tai 1. Välittäjältä pyydetään uutta puhdasta yhteyttä asettamalla cleanSession-

lippumuuttujan arvoksi 1, kun taas arvolla 0 välittäjä käyttää pysyvää yhteyttä ja säilöö edellä mainitut istuntotiedot.

CONNECT-paketti on yhteyden alussa lähetetty paketti, joka sisältää tarvittavat tiedot yhteyden muodostamiseksi. CONNECT-paketti koostuu useista tavuista, joista yksi tavu on varattu yhteyden määrittäville lippumuuttujille (Taulukko 1).

Taulukko 1. Connect-paketin tavu, jossa lippumuuttujien arvot määritellään.

Bittinumero	**Lippumuuttuja**
7	Käyttäjätunnus (User Name Flag)
6	Salasana (Password Flag)
5	Tahtoviestin säilytys (Will Retain)
4	Tahtoviestin laatutaso (Will QoS)
3	Tahtoviestin laatutaso (Will QoS)
2	Tahtoviesti (Will Flag)
1	Puhdas istunto (cleanSession)
0	Varattu (Reserved)

Tietotekniikassa tavun jokainen bitti voi saada arvon False/0 tai True/1, joten puhdasta istuntoa aloittaessa tulee 1. bitille asettaa arvoksi True/1 ja pysyvää istuntoa käytettäessä bitin arvoksi asetetaan False/0. Puhdas istunto on istunto, jossa istuntojen välillä tietoja ei säilytetä, kuten pysyvässä istunnossa. [20, s. 30–32.]

4.3.4 Yhteydenhallinta ja katkeamisten käsittely

Viimeinen tahto ja testamentti

MQTT-protokollassa on mahdollista määrittää asiakkaalle viimeinen tahto ja testamentti (Last Will and Testament – LWT), joka on viesti, joka lähetetään, mikäli havaitaan odottamaton yhteyden katkeaminen asiakkaan ja välittäjän välillä. Kun yhteys katkaistaan suunnitellusti asiakas lähettää DISCONNECT-paketin välittäjälle, joten yhteyden katkeaminen ilman DISCONNECT-pakettia tulkitaan odottamattomaksi katkeamiseksi. [21.]

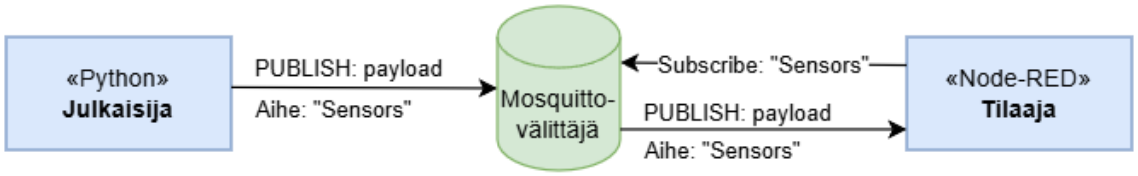
Tahtoviesti määritellään yhteyden muodostuksen yhteydessä CONNECT-paketissa ja viestiä määritettäessä sille asetetaan aihe, viestien välityksen laatutaso, tahtoviestin säilytys sekä itse viestin sisältö. Odottamattomassa katkeamistilanteessa viesti välittyy viestien välityksen laatutason mukaisesti kaikille niille muille asiakkaille, jotka ovat tilanneet sen aiheen, joka sisältää kyseisen tahtoviestin. [21.]

Asiakasyhteyksien ylläpito

Asiakasyhteyden ylläpito (Keep Alive) on MQTT-protokollan ominaisuus, jolla välittäjä tarkkailee yhteyksien tilaa sekä havaitsee, mikäli yhteys johonkin asiakaslaitteeseen on menetetty. CONNECT-paketissa voidaan määrittää yhteydelle yhteyden ylläpidon tarkastusväli, joka on suurin aika, jonka asiakas voi olla ottamatta yhteyttä välittäjään. Jos asiakas ei lähetä mitään viestiä tarkkailuintervallin kuluessa, sen on lähetettävä PINGREQ-paketti ilmaistakseen, että yhteys on vielä ehjä. Välittäjä vastaa PINGREQ-pakettiin omalla PINGRESP-paketillaan. Mikäli välittäjälle ei saavu mitään viestejä määrätyssä ajassa, se sulkee yhteyden ja lähettää mahdollisesti määritetyt asiakkaan viimeiset tahtoviestit. [22.]

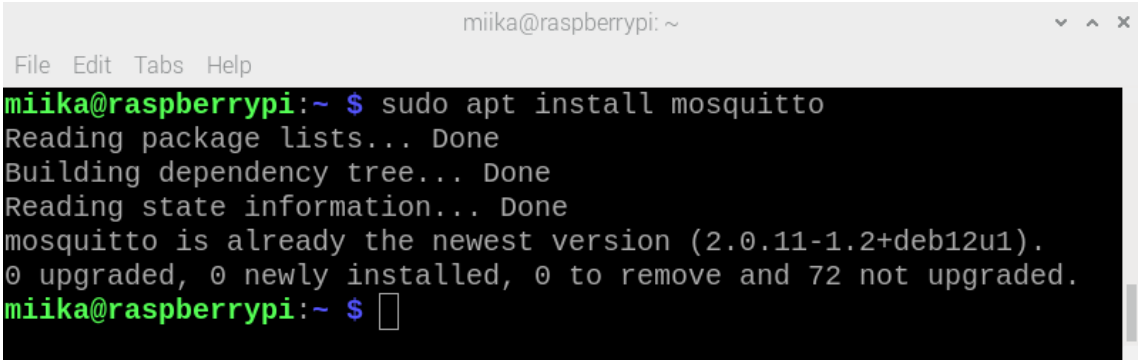
4.4 MQTT-protokollan käyttöönotto

Tässä luvussa käsitellään, kuinka Mosquitto-välittäjäohjelmisto otetaan käyttöön ja kuinka Python-ohjelmasta saadaan lähetettyä viestejä MQTT-välittäjälle Paho-kirjastoa käyttämällä. Ympäristöolosuhteiden mittausdata luetaan ja esikäsitellään käyttäen Python-ohjelmaa, josta tieto halutaan siirtää Node-RED-ohjelman käyttöön. MQTT-protokollan käyttö edellyttää, että järjestelmässä tulee olla yksi välittäjä sekä kaksi asiakasta (Kuva 9), joista toinen toimii julkaisijana osana Python-ohjelmaa ja toinen tilaajana osana Node-RED-ohjelmaa. Kaikkia kolmea ohjelmistoa ajetaan samalla Raspberry Pi:llä.



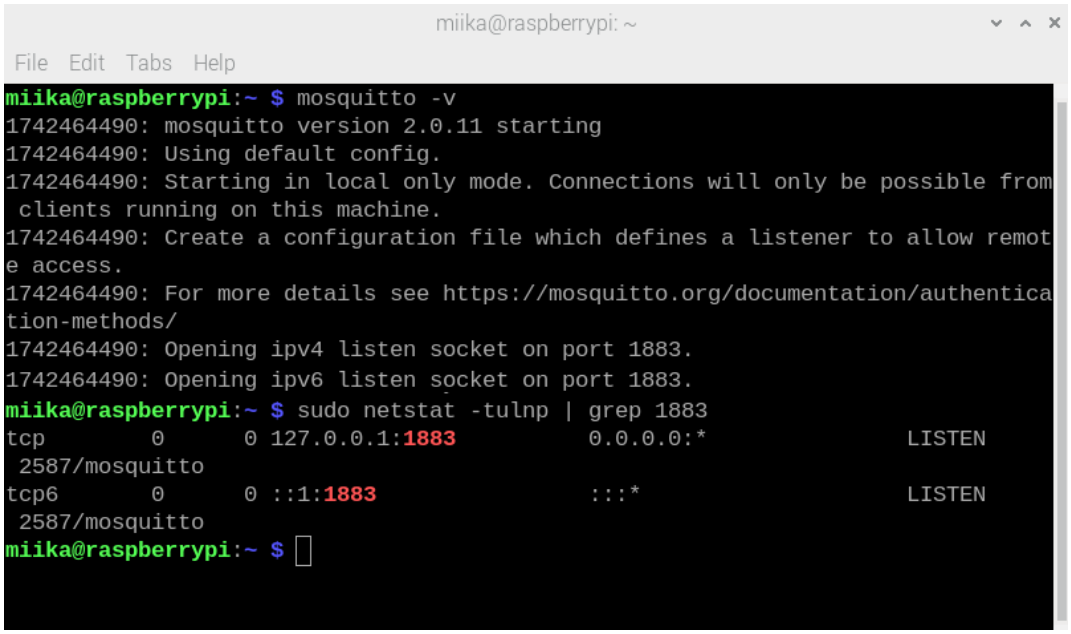
Kuva 9. MQTT-protokollan mukainen arkkitehtuuri.

MQTT-välittäjänä käytetään paikallisesti asennettua Eclipse Foundationin avoimen lähdekoodin Mosquitto-välittäjää. Mosquitto-välittäjäohjelmisto asennetaan Linux-ympäristössä APT-paketinhallintaohjelmiston (Advanced Package Tool) avulla terminaalikomentoja käyttämällä (Kuva 10). Ennen Mosquitton asentamista Raspberry Pi:n pakettilistan tiedot tulisi päivittää ajantasaisiksi paketinhallintaohjelmistolla käyttäen komentoa ”sudo apt update” [23].



Kuva 10. Eclipse Mosquitto välittäjäohjelmiston asentaminen.

Nyt Mosquitto on asennettuna paikallisesti Raspberry Pi:lle. Se on ajettavissa käyttöjärjestelmän komentorivin kautta komennolla ”mosquitto -v” (Kuva 11), jolloin välittäjä käynnistyy ja kuuntelee Raspbery Pi:n oletusporttia 1883 sinne saapuvan viestiliikenteen varalta.



Kuva 11. Mosquitto-välittäjä ohjelmiston käynnistäminen.

Mosquitto-välittäjän käynnistyessä se avaa oletusporttiin 1883 kaksi viestiliikenteen kuuntelurajapintaa, joista toinen kuuntelee IPv4-protokollan, eli Internet Protokolla version 4 mukaista liikennettä ja toinen IPv6-prokollan mukaista liikennettä. Terminaalikomennolla ”sudo netstat -tulnp | grep 1883” tarkastetaan, mitkä ohjelmistot käyttävät 1883-porttia. Näin varmistutaan, että portti on Mosquitton käytössä.

Python-julkaisija

Python-asiakasohjelmistona tässä työssä käytetään Pythonin paho-mqtt-kirjastoa ja sen funktioita. MQTT-kirjaston asentamisessa käytetään pip-paketinhallintaohjelmistoa, jolloin paho-mqtt-kirjaston asentaminen Raspberry Pi:lle

tapahtuu terminaalikomennolla ”pip install paho-mqtt” [23]. Kun paho-mqtt-kirjasto on asennettu laitteelle, se täytyy tuoda Python-ohjelman käyttöön lisäämällä kirjastoon viittaava import-komento opinnäytetyön luvussa 3.3 esitellyn ohjelmakoodin alkuun (Kuva 12).

```
1  import sys
2  import json
3  import paho.mqtt.client as paho
4  import smbus
5  import time
6  import dhtlib
7
8  address = 0x48
9
10 A0 = 0x40
11 bus = smbus.SMBus(1)
12 dhtnum = str(11)
13 dhtpin = str(4)
14
15 client = paho.Client()
16
17 if client.connect("localhost", 1883, 60) != 0:
18     print("Couldn't connect to the mqtt broker")
19     sys.exit(1)
20
21 while True:
22     bus.write_byte(address,A0)
23     value = bus.read_byte(address)
24     msg = dhtlib.start_read(dhtnum, dhtpin)
25     msg["brightness"] = value
26     payload = json.dumps(msg)
27     client.publish("sensors", payload)
28     time.sleep(5)
29
```

Kuva 12. Python-ohjelma ja sen sisältämät mittauksien lukemiset ja MQTT-protokollan mukainen mittaviestien julkaiseminen.

Paho-kirjastoa käyttämällä muodostin luo olio-ohjelmoinnin mukaisen asiakasolion koodirivillä 15, jonka jälkeen olio-objektin ja paikallisesti ajetun Mosquitto-välittäjän välille luodaan yhteys koodirivillä 17. Välittäjään yhteyttä luotaessa connect-funktio saa kolme parametria ”localhost”, 1883 ja 60.

- IP-osoite/Isäntäkoneen nimi: ”Localhost” määrittää, että MQTT-välittäjä, johon yhteyttä muodostetaan, sijaitsee samalla laitteella kuin Python-ohjelma.

- Portin numero: Määritetään, että portti, johon otetaan yhteyttä, on 1883.

- Yhteyden tarkastusväli: Yhteyden elossa oloa tarkkaillaan 60 sekunnin välein.

Python-pääohjelman lopussa rivillä 27 käytetään publish-funktiota viestin lähettämiseen välittäjälle, jolloin argumentteina annetaan viestin aihe sekä viestisisältö, eli tämän ohjelman tapauksessa muuttuja, jonka nimi on payload.

5 Node-RED:in ja pilvitietokannan integraatio

Tässä luvussa käsitellään Node-RED-ohjelmointityökalua ja MongoDB Atlas -pilvitietokantaa ja näiden käyttöönottoa. Luvussa käydään läpi Node-RED:in asennusprosessi, kuinka sillä ohjelmoidaan ja vastaanotetaan viestejä MQTT-välittäjältä. Luvun lopussa käydään läpi kuinka Node-RED-ohjelmointityökalu ja MondoDB Atlas -pilvitietokanta yhdistetään toisiinsa.

5.1 Node-RED:in käyttöönotto

Node-RED on avoimen lähdekoodin graafinen ohjelmointiympäristö, jota kehittää OpenJS Foundation. Sen kehittäminen alkoi vuonna 2013 ja alkujaan tarkoituksena oli luoda työkalu MQTT-aiheiden visualisointiin ja hallintaan. Node-RED:in käyttötarkoitukset ovat monipuolistuneet alkuperäisestä suunnitelmasta. Nykyään Node-RED on yleiskäyttöinen työkalu kevyiden tapahtumapohjaisten sovelluksien rakentamiseen. Node-RED on selainpohjainen Node.js:ään perustuva ohjelmointiympäristö, joka hyödyntää valmiita noodeja (nodes), joita yhdistämällä luodaan Node-RED:in työkiertoja ja datavirtoja (flows). Node.js on alustariippumaton ajoympäristö JavaScipt-koodin ajamista varten. Saatavilla on yli 5 000 erilaista valmista noodia, jotka mahdollistavat monipuoliset integraatiot ja automaatiot. Node-RED:in selaineditori mahdollistaa myös omien JavaScript-funktioiden tekemisen. [24.]

Node-RED:in asentaminen

Raspberry Pi:llä Node-RED-ympäristön käyttöönotto on nopea prosessi. Node-RED, npm eli JavaScript-kielen paketinhallintatyökalu ja Node.js saadaan Github-verkkosivulta haettua ja asennettua komentosarjalla 1 käyttöjärjestelmän komentorivin kautta [25].

```
bash <(curl -sL https://raw.githubusercontent.com/node-red/linux-installers/master/deb/update-nodejs-and-nodered)
```

Esimerkkikoodi 1. Node-RED, npm ja Node.js asennus terminaalin kautta.

Kun Node-RED:in käyttöä varten tarvittavat ohjelmistot ovat asennettu, voidaan Node-RED käynnistää joko paikallisesti terminaalissa tai asettaa Node-RED toimimaan palveluna, jolloin ohjelmisto pyörii taustalla eikä terminaalin kautta. Komentosarjalla 2 Node-RED voidaan käynnistää palveluna ja konfiguroida käynnistymään automaattisesti Raspberry Pi:n käynnistyessä [25].

```
node-red-start
sudo systemctl enable nodered.service
```

Esimerkkikoodi 2. Node-RED:in käynnistäminen palveluna ja automaattikäynnistyksen valitseminen.

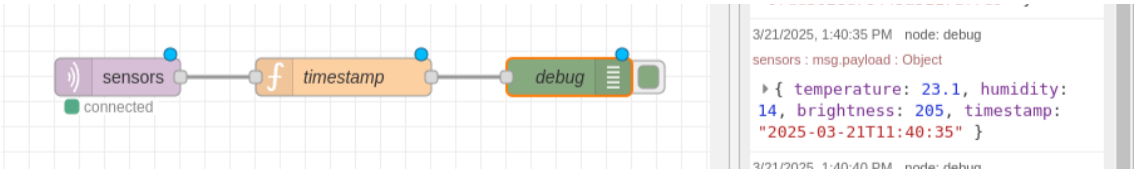
Node-RED käyttää oletuksena porttia 1880, joten sen selaineditori saadaan auki kirjoittamalla selaimen osoiteriville ”http://<hostname>:1880”, jossa hostname korvataan joko isäntälaitteen nimellä tai IP-osoitteella. Samalla laitteella käynnissä olevaan Node-RED-selaineditoriin päästään kirjoittamalla selaimen osoiteriville ”http://localhost:1880”. [25.]

5.2 MQTT-aiheen tilaaminen Node-RED:illä

Aiheen tilaamista varten tarvittavat noodit vedetään selaineditorin vasemmassa laidassa sijaitsevasta paletista työtasolle niiden käyttöä varten. Mikäli aiheen tilaamiseen tai muuhun haluttuun käyttötarkoitukseen tarvittavia noodeja ei löydy paletista, niitä voidaan asentaa lisää ylävalikosta löytyvän Manage pallet -

valikosta. Noodien asetuksia ja konfiguraatioita päästään tarkastelemaan noodia kaksoisklikkaamalla, ja noodit yhdistetään toisiinsa hiirellä vetämällä.

Ympäristöolosuhteiden mittausdatan saamiseksi Python-ohjelmistolta Node-RED:ille konfiguroidaan MQTT input -noodi, jonka kautta aiheen tilaaminen tapahtuu. MQTT-noodin perään lisätään funktio-noodi, johon kirjoitetaan JavaScript-funktio lisäämään aikaleima jokaiseen saapuvaan viestipakettiin. Viimeiseksi asetetaan debug-noodi, joka mahdollistaa saapuvien pakettien tarkkailun selaineditorissa (Kuva 13).



Kuva 13. MQTT-aiheen tilaamiseen ja aikaleimaamiseen tarkoitettu työkierto Node-RED:issä.

Ympäristöolosuhteiden mittaviestit välittyvät nyt Python-ohjelmalta välittäjän kautta Node-RED:illä toimivalle tilaajalle ja debug-noodilla huomataan, että viestit näyttävät sisällöltään olevan oikeassa formaatissa ja aikaleimaantuvat halutusti.

5.3 SQL- ja NoSQL-tietokannat

Yleisimmin käytettyjä tietokantatyyppejä ovat SQL- ja NoSQL-tietokannat. SQL eli Structured Query Language on tietokantojen käyttämiseen erikoistunut kyselykieli. Tietokantaa, jonka pääasiallinen käyttötyökalu on SQL, kutsutaan SQL-tietokannaksi. SQL-tietokannat ovat relaatiotietokantoja, joissa tieto tallennetaan riveistä ja sarakkeista muodostuviin tauluihin. Taulujen välisiä yhteyksiä määritetään käyttämällä viiteavaimia. Relaatiotietokannat luodaan ja hallitaan ennalta määrätyn tietorakenteen mukaan, jonka vuoksi relaatiotietokannat eivät kykene käsittelemään jäsentämätöntä dataa, joka ei ole määrätyn tietomallin mukaista. [26.]

Toisin kuin SQL-tietokannat NoSQL-tietokannat (Not only SQL) on suunniteltu käsittelemään myös jäsentämätöntä ja puolistrukturoitua dataa ei-taulukkomaisella tallennusmallilla. NoSQL-tietokantoihin dataa voidaan tallentaa sen alkuperäisessä muodossa ilman, että sen täytyy noudattaa ennalta määrättyä rakennetta. Tämä mahdollistaa esimerkiksi kuvien, videoiden ja teksidokumenttien tallentamisen, mikä ei ole mahdollista perinteisissä relaatiotietokannoissa. Eri NoSQL-tietokannat tallentavat tietoa eri tavalla, kuten dokumentteina, avain-arvopareina tai graafeina. Tässä työssä käytetty tietokanta on malliltaan dokumenttitietokanta, joka tallentaa tiedot JSON-objekteja muistuttavina dokumentteina. [26.]

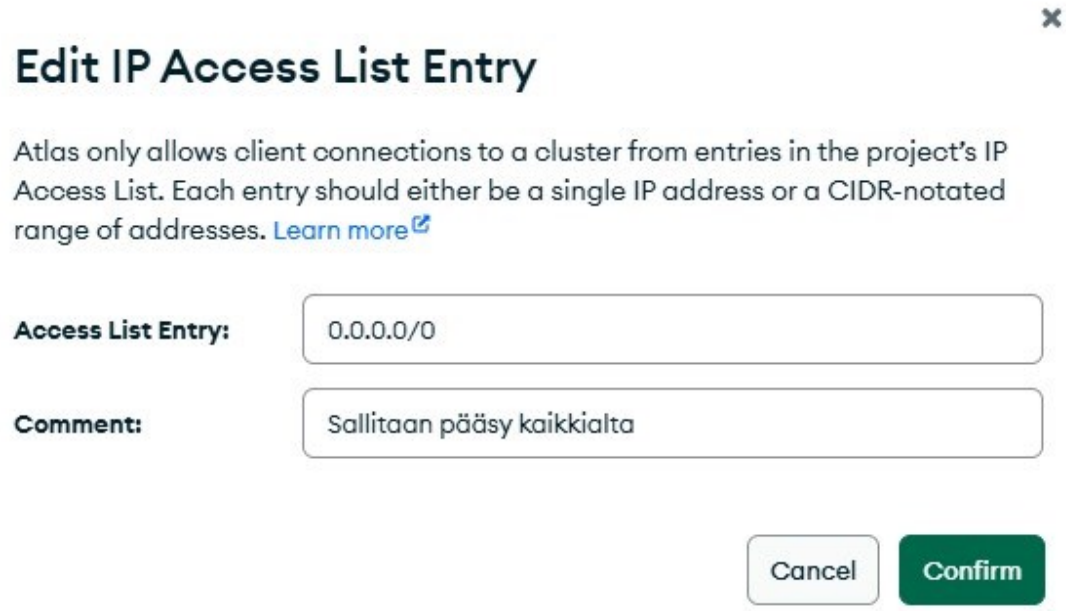
5.4 MongoDB Atlaksen käyttöönotto

MongoDB Atlas on MongoDB:n NoSQL-pilvitietokanta, jota tarjotaan DBaaS-periaatteella (Database-as-a-Service). Tämä tarkoittaa sitä, että pilvitietokannan ylläpitäjä vastaa tietokannan laitteistoista, ohjelmistopäivityksistä ja muusta ylläpidosta, jolloin käyttäjän ei tarvitse keskittyä tietokantainfrastruktuurin ylläpitämiseen. [27.]

Käyttöönotto on tehty käyttäjälle nopeaksi. Kun palveluun on luotu käyttäjätili ja määritelty organisaatio, projekti ja muut alkumäärittelyt, tulee seuraavaksi luoda klusteri. Klusterin luominen tapahtuu käyttöliittymän valikosta Data storage – Clusters -valikosta. Klusteria luodessa määritetään, otetaanko käyttöön jaettu klusteri, dedikoitu klusteri vai monipilvi- ja monialueklusteri. Klusterityyppi määrittää sen, miten ja missä dataa säilytetään. Otetaan mittadatan säilömistä varten käyttöön ilmainen jaettu klusteri, jolloin data säilötään muiden kanssa jaetuille palvelimille. [27.] Hierarkkisesti klusterin alapuolelle MondoDB:ssä määritetään vielä itse tietokanta, jonka sisälle puolestaan voidaan luoda yksi tai useampi kokoelma. Nimetään tietokanta nimellä ”Opinnäytetyö” ja luodaan yksi kokoelma nimellä ”Anturointipaketti1”.

Tietokantaan yhdistäminen

Tietokannan käyttöönottamiseksi on määriteltävä ne verkon osoitteet, joista tietokannan käyttäminen on sallittu (Kuva 14).

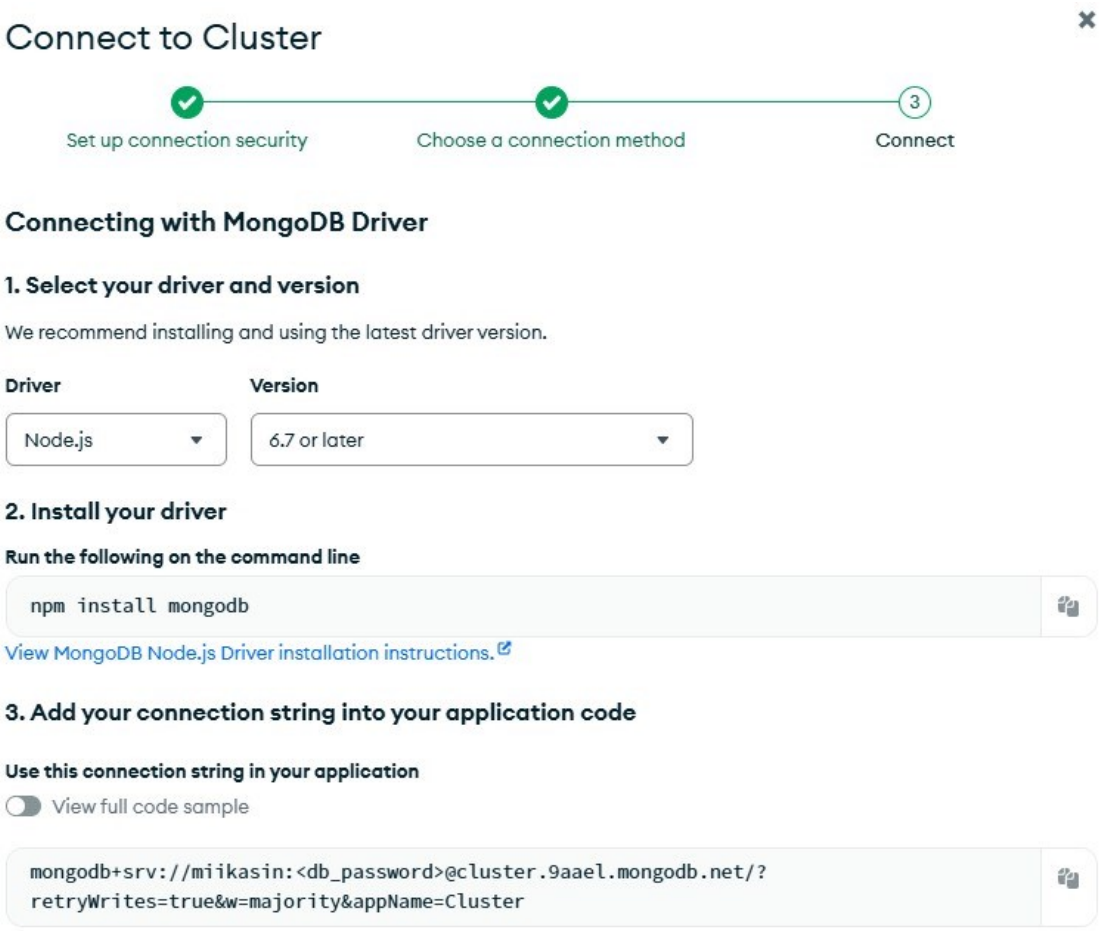


Kuva 14. Pilvitietokantaan pääsyn salliminen määritetystä IP-osoitteesta.

Pilvitietokannan turvallisuusmäärittelyissä voidaan sallia pääsy kaikkialta verkosta, vain tietystä aliverkosta tai vain tietystä IP-osoitteesta. Kuvan määrittely ”0.0.0.0/0” sallii yhteyden kaikista IPv4-osoitteista. Kun tietokantaa halutaan käyttää toisen ohjelman kautta, on luotava tietokannalle käyttäjä, jolla on riittävästi käyttöoikeuksia suunniteltua käyttötarkoitusta varten. Tietokantaan lisätään uusia käyttäjiä ja käyttöoikeusryhmiä valikon ”Security – Database Access” kautta. [27.]

Tietokannan ottamiseksi ulkoisen sovelluksen käyttöön on luotava yhteysmerkkijono, jonka avulla sovellus ottaa yhteyden tietokantaan. Yhteysmerkkijonon luominen tapahtuu klusterin pääsivulla sijaitsevan connect-napin kautta. Valikosta valitaan vaihtoehto yhdistää tietokantaan MondoDB:n natiiviajureita

käyttäen. [27.] Avautuvasta valikosta valitaan ajuriksi ”Node.js” versio 6.7 (Kuva 15), jotta tietokanta saadaan Node-RED:in käyttöön.



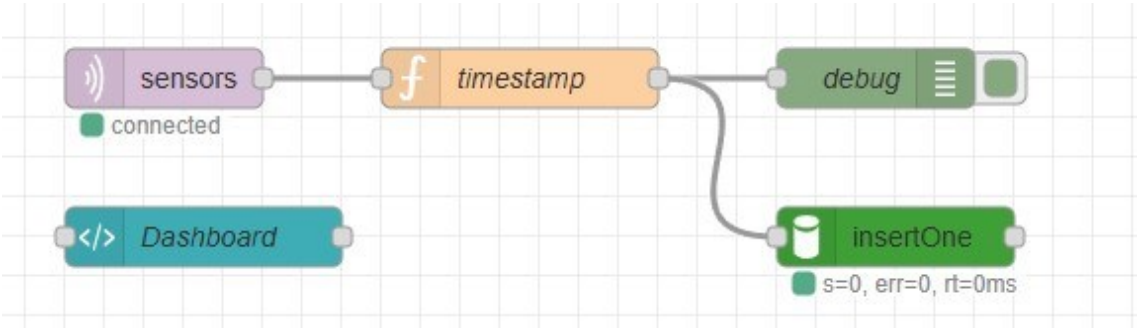
Kuva 15. Ajurin valinta ja yhdistysmerkkijonon luominen.

Yhdistysmerkkijono sisältää paikan tietokantakäyttäjälle ja käyttäjän salasanalle. Tietokantaan autentikointia varten tähän asetetaan halutun käyttäjätilin tiedot.

5.5 Tietokannan Node-RED-integraatio

MongoDB Atlas -tietokannan käyttämiseksi Node-Red:iin tulee asentaa mongodb4-paketti, joka sisältää MongoDB Atlaksen käyttöön tarvitut noodit. MongoDB-noodeista käytetään InsertOne-noodia (Kuva 16), joka lisää yksittäisen dokumentin tietokantaan. Määritetään noodi ottamaan yhteyttä haluttuun

tietokantaan Advanced Connection URI -valikon kautta. URI-kenttään kopioidaan aikaisemmin luotu yhteysmerkkijono, jonka lisäksi määritetään tietokannan nimi, kokoelman nimi sekä klusterikäyttäjän käyttäjänimi ja salasana (liite 2). [28.]

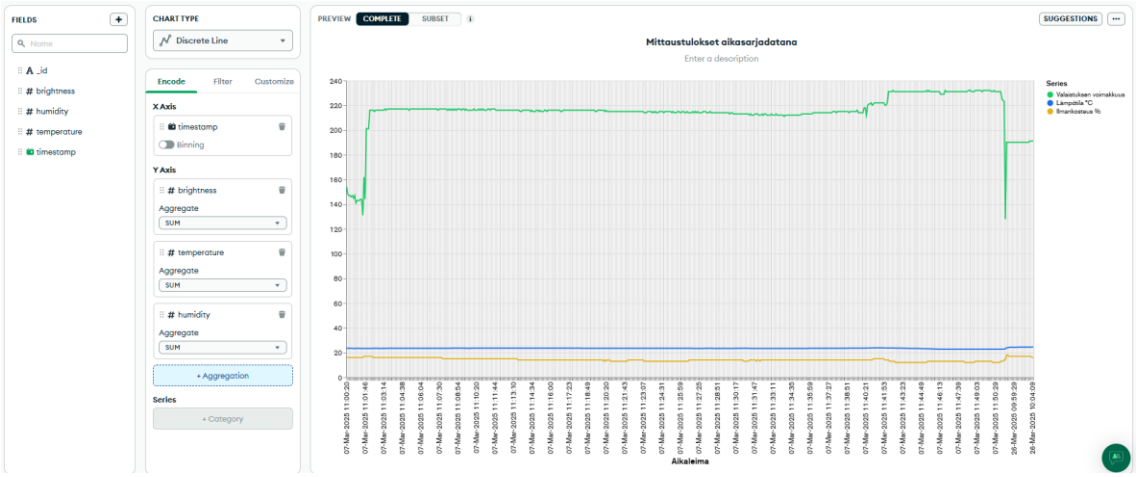


Kuva 16. Node-RED:in työkierto, joka sisältää viestin vastaanottamisen ja tietokantaan viemisen.

Python-ohjelmalta saapunut aikaleimattu viesti yhdistetään InsertOne-solmuun, jolloin mittausviestin tallentaminen tietokantaan on tapahtumapohjaista ja tapahtuu aina silloin, kun uusi viesti vastaanotetaan.

Datan visualisointi Node-RED:issä

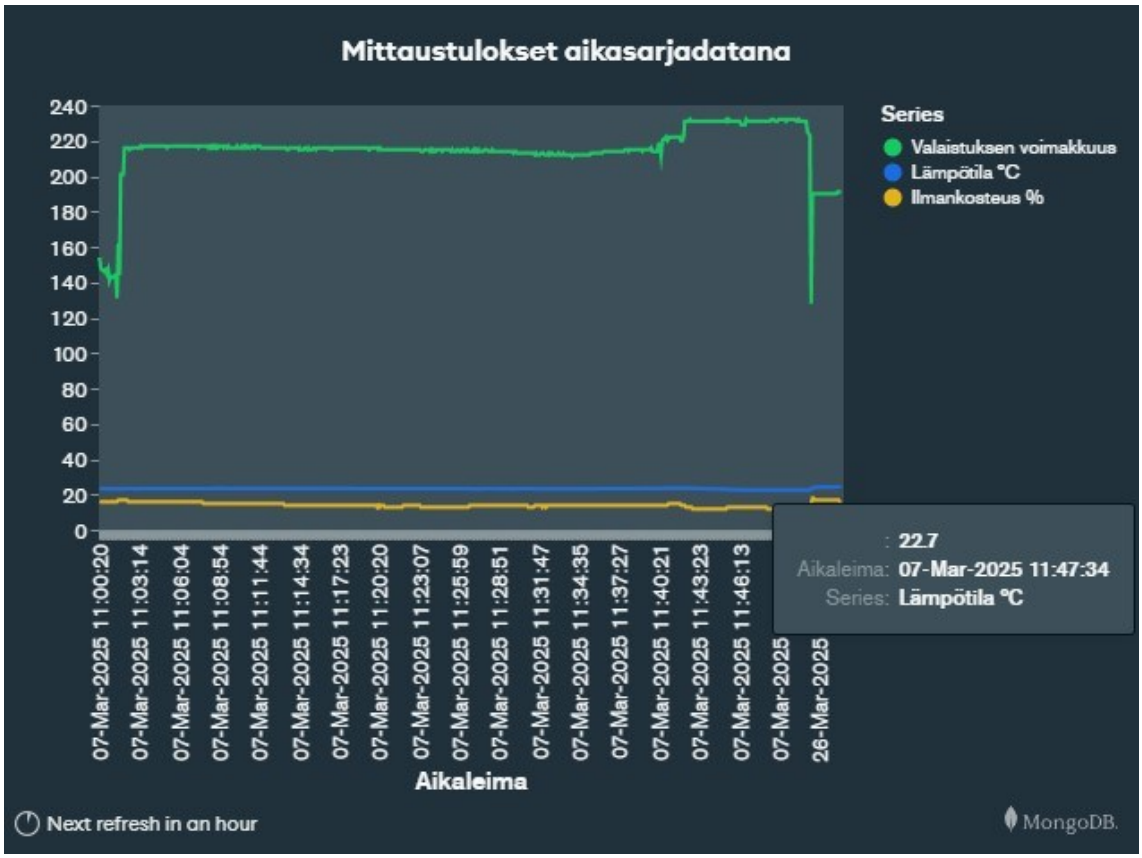
Ympäristöolosuhteiden mittaustulokset visualisoidaan MongoDB Atlas -tietokannan työkaluilla, jonka jälkeen visualisointi upotetaan osaksi Node-RED-ohjelmaa. Kuvaajat laaditaan siirtymällä tietokannan käyttöliittymän Charts-välilehdelle, josta löytyy Add Dashboard -painike. Kuvaajaa laadittaessa valitaan aukeavasta valikosta kuvaajan datalähteeksi Anturointipaketti1-kokoelma, johon kaikki ympäristöolosuhteiden mittaustiedot on tallennettu. Työkalu tunnistaa eri mittasuureet avain-arvo-parien perusteella ja tuo kaikki suureet näkyville työkalusivun oikeaan laitaan (Kuva 17).



Kuva 17. MongoDB Atlaksen kuvaajien piirtoon tarkoitettu työkalu.

Asetetaan x-akselille aikaleima ja mitatut suureet y-akselille samaan kuvaajaan. Aikaleimalle valitaan selkeä formaatti ja suureille annetaan kuvaavat nimet.

Node-RED-työpöydällä valitaan node-red-dashboard-pakettiin kuuluva template-noodi, jonka avulla kuvaaja tuodaan sovelluksen käyttöön. Template-noodi näkyy irtonaisena työpöydällä kuvassa 18. MongoDB Atlaksen käyttöliittymässä kuvaajan asetuksista valitaan ”Embed” ja tavaksi ”iFrame”. Näin saadaan luotua kuvaajan upottamiseen tarvittava merkkijono. Merkkijono liitetään template-noodin määrittelykenttään, jonka jälkeen kuvaaja on saatu Node-RED-ohjelman käyttöön (Kuva 18).



Kuva 18. Node-RED:in visualisointinäkymä mittausdatalle.

MongoDB Atlaksella laadittu ympäristöolosuhteiden kuvaaja piirtää mittaustulokset aikasarjana ja hakee automaattisesti ajantasaiset tiedot tietokannasta kuvaajan asetuksissa määritetyllä päivitystiheydellä.

6 Yhteenveto

Insinöörityössä käsiteltiin ympäristöolosuhteiden mittaamista, mittadatan siirtämistä MQTT-protokollan avulla ohjelmalta toiselle ja NoSQL-tietokantaan tallentamista. Insinöörityön tavoitteena oli saada valmis IoT-laitteisto mittadatan keräämistä varten, joka voisi toimia perustana pienille tai keskisuurille automaatiosovelluksille.

Mittauksiin vaadittavien komponenttien ja ohjelmistojen valinta onnistuivat, ja työn lopputulos oli odotusten mukainen kokonaisuus. DHT11-anturin ja uusimman Raspberry Pi -mallin välisistä kommunikaatiohaasteista opittiin, kuinka GPIO-pinnejä voidaan lukea käyttämättä yleisimpiä valmiita Python-kirjastoja. MQTT-protokollan eri ominaisuuksia ja mahdollisuuksia IoT-laitteistoissa käsiteltiin sopivasti siinä laajuudessa, kuin työn tekemisessä oli tarpeen.

Tässä työssä kehitettyä kokonaisuutta voidaan jatkokehityksessä laajentaa lisäämällä hajautetusti mikrokontrolleja, joille asennetaan lisää MQTT-asiakasohjelmistoja. Tällöin Raspberry Pi toimisi MQTT-välittäjän osana laajempaa hajautettua järjestelmää. Mikrokontrollereja voitaisiin käyttää erilaisten laitteiden ohjaamiseen mittaustuloksien perusteella. Selvitystyötä liittyen ympäristöolosuhteiden mittaamiseen ja säilömiseen voitaisiin jatkaa vertailemalla NoSQL- ja SQL-tietokantojen eroavaisuuksia mittadatan säilömisessä.

Lähteet

1 What is a Raspberry Pi? Verkkoaineisto. Opensource. <https://opensource.com/resources/raspberry-pi>. Luettu 17.2.2025.

2 Raspberry Pi hardware. Verkkoaineisto. Raspberry Pi Foundation. <https://www.raspberrypi.com/documentation/computers/raspberry-pi.html>. Luettu 17.2.2025.

3 Raspberry Pi OS Overview. 2020. Verkkoaineisto. PiCockpit. <https://picockpit.com/raspberry-pi/raspberry-pi-os-overview/>. 10.11.2020. Luettu 18.2.2025.

4 Wolfgang, Männel. 2020. The beginner's guide to working with the Terminal on the Raspberry Pi from a Windows, macOS, or Linux computer. Verkkoaineisto. TheDigitalPictureFrame. <https://www.thedigitalpictureframe.com/beginners-guide-terminal-raspberry-pi-windows-macos-or-linux/>. Päivitetty 3.12.2023. Luettu 18.2.2025.

5 Rana, Amit. 2019. How LDR Works (Light Dependent Resistor). Verkkoaineisto. Spectra One. <https://kitflix.com/how-ldr-works-light-dependent-resistor>. 15.5.2019. Luettu 19.2.2025.

6 DHT-Temperature and Humidity Sensor. 2021. Verkkoaineisto. Components101. <https://components101.com/sensors/dht11-temperature-sensor>. 16.7.2021. Luettu 19.2.2025.

7 Yu, Changxin. Temperature and Humidity Sensors. Verkkoaineisto. Duke MEMS. <https://sites.duke.edu/memscapstone/temperature-and-humidity-sensors/>. Luettu 20.2.2025.

8 DHT11 Humidity & Temperature Sensor. Verkkoaineisto. Mouser Electronics. <https://www.mouser.com/datasheet/2/758/DHT11-Technical-Data-Sheet-Translated-Version-1143054.pdf?srsltid=AfmBOoqn5wN87-NQLQHFFGMi5eXsB7uHrHp7T7LVETc6tf7OrwxHF8By>. Luettu 19.2.2025.

9 Using the I2C Bus. Verkkoaineisto. Robot Electronics. <https://www.robot-electronics.co.uk/i2c-tutorial>. Luettu 25.2.2025.

10 Elecrow Raspberry Pi & Arduino Starter Kit. Käyttöohje. Elecrow.

11 Szarvas, Zoltan. Verkkoaineisto. Egalnetsoftwares. < https://www.egalnetsoftwares.com/_data/raspcontroller/dht_v6.py>. Luettu 28.2.2025.

12 Acsany, Philipp. Working With JSON Data in Python. 2024. Verkkoaineisto. Real Python. <https://realpython.com/python-json/>. 22.12.2024. Luettu 26.2.2025.

13 What is MQTT? Verkkoaineisto. Amazon Web Services. <https://aws.amazon.com/what-is/mqtt/>. Luettu 26.2.2025.

14 HiveMQ Team. A Beginner’s Guide to MQTT Brokers. 2023. Verkkoaineisto. HiveMQ. <https://www.hivemq.com/blog/mqtt-brokers-beginners-guide/>. 27.11.2023. Luettu 26.2.2025.

15 Learn MQTT Broker. Verkkoaineisto. Catchpoint. <https://www.catchpoint.com/network-admin-guide/mqtt-broker>. Luettu 3.3.2025.

16 HiveMQ Team. MQTT Client, MQTT Broker, and MQTT Server Connection Establishment Explained – MQTT Essentials: Part 3. 2023. Verkkoaineisto. HiveMQ. <https://www.hivemq.com/blog/mqtt-essentials-part-3-client-broker-connection-establishment/>. 7.6.2023. Luettu 3.3.2025.

17 HiveMQ Team. MQTT Topics, Wildcards, & Best Practices – MQTT Essentials: Part 5. 2024. Verkkoaineisto. HiveMQ. <https://www.hivemq.com/blog/mqtt-essentials-part-5-mqtt-topics-best-practices/>. 20.2.2024. Luettu 4.3.2025.

18 HiveMQ Team. What is MQTT Quality of Service (QoS) 0,1, & 2? – MQTT Essentials: Part 6. 2024. Verkkoaineisto. HiveMQ. <https://www.hivemq.com/blog/mqtt-essentials-part-6-mqtt-quality-of-service-levels/>. 20.2.2024. Luettu 4.3.2025.

19 HiveMQ Team. Understanding Persistent Sessions and Clean Sessions – MQTT Essentials: Part 7. 2024. Verkkoaineisto. HiveMQ. <https://www.hivemq.com/blog/mqtt-essentials-part-7-persistent-session-queuing-messages/>. 21.2.2024. Luettu 5.3.2025.

20 MQTT Version 5.0 OASIS Standard. 2019. Verkkoaineisto. OASIS Open. <https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.pdf>. 7.3.2019. Luettu 27.2.2025.

21 HiveMQ Team. What is MQTT Last Will and Testament (LWT)? – MQTT Essentials: Part 9. 2024. Verkkoaineisto. HiveMQ. <https://www.hivemq.com/blog/mqtt-essentials-part-9-last-will-and-testament/>. 21.2.2024. Luettu 6.3.2025.

22 HiveMQ Team. What Is MQTT Keep Alive and Client Take-Over? – MQTT Essentials Part 10. 2024. Verkkoaineisto. HiveMQ. <https://www.hivemq.com/blog/mqtt-essentials-part-10-alive-client-take-over/>. 21.2.2024. Luettu 6.3.2025.

23 Potekhina, Anastasia. A Beginner’s Guide to MQTT: Understanding MQTT, Mosquitto Broker, and Paho Python MQTT Client. 2023. Verkkoaineisto. Medium. <https://medium.com/@potekh.anastasia/a-beginners-guide-to-mqtt-understanding-mqtt-mosquitto-broker-and-paho-python-mqtt-client-990822274923>. 20.6.2023. Luettu. 7.3.2025

24 About. Verkkoaineisto. Node-RED. <https://nodered.org/about/>. Luettu 10.3.2025.

25 Running on Raspberry Pi. Verkkoaineisto. Node-RED. <https://nodered.org/docs/getting-started/raspberrypi>. Luettu 10.3.2025.

26 Understanding SQL vs NoSQL Databases. Verkkoaineisto. MongoDB. <https://www.mongodb.com/resources/basics/databases/nosql-explained/nosql-vs-sql>. Luettu 27.3.2025.

27 MongoDB Atlas Tutorial. Verkkoaineisto. MongoDB. <https://www.mongodb.com/resources/products/platform/mongodb-atlas-tutorial>. Luettu 17.3.2025.

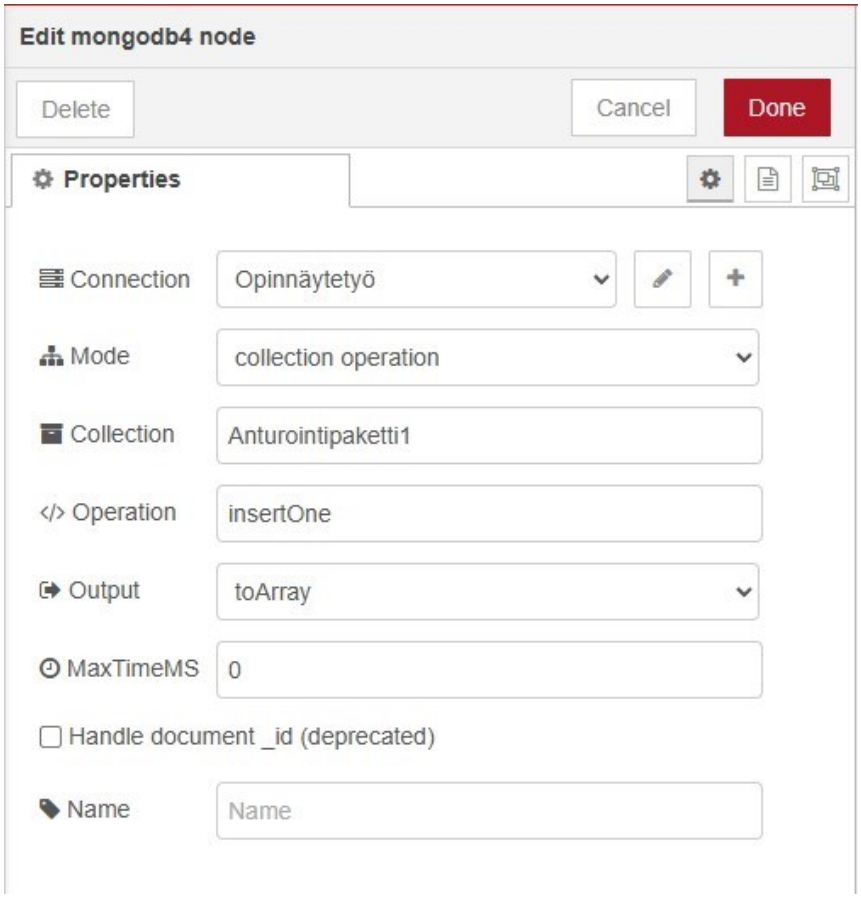
28 Using MongoDB With Node-RED. Verkkoaineisto. FlowFuse. <https://flowfuse.com/node-red/database/mongodb/>. Luettu 18.3.2025.

DHT-kirjaston muokattu loppuosuus

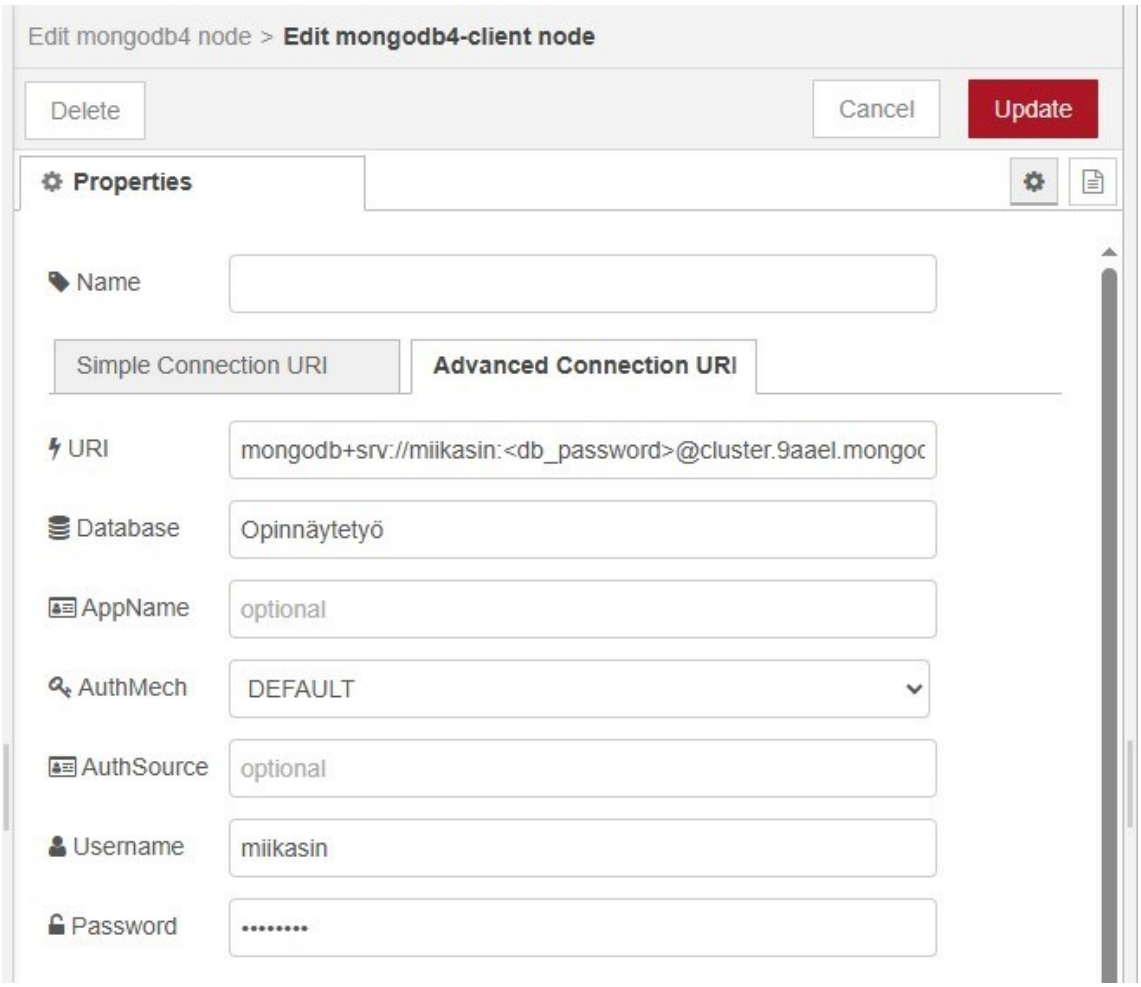
```
242 def start_read(dht, pin):
243     # Parse command line parameters
244         sensor = dht
245         if sensor not in ("11", "22", "2302"):
246             print('Sensor "{}" is not valid. Use 11, 22 or 2302'.format(sensor))
247             exit(1)
248         isDht11 = sensor == "11"
249         try:
250             pin = int(pin)
251             if (pin < 2 or pin > 27):
252                 raise ValueError
253         except:
254             print('Gpio {} is not valid'.format(pin))
255             exit(1)
256 
257         # initialize GPIO
258         GPIO.setwarnings(False)
259         GPIO.setmode(GPIO.BCM)
260 
261         # read data
262         MAX_ATTEMPTS = 15
263         MAX_NOT_FOUND_ATTEMPTS = 3
264         dht = DHT(pin, isDht11)
265         result = dht.read()
266         not_found_attempts = 0
267 
268         # make some attempts, because someone may not be successful
269         for x in range(0, MAX_ATTEMPTS):
270             if result.is_valid() or not_found_attempts == MAX_NOT_FOUND_ATTEMPTS:
271                 break
272             else:
273                 time.sleep(2)
274                 result = dht.read()
275                 if result.error_code == DHTResult.ERR_NOT_FOUND:
276                     not_found_attempts += 1
277                 else:
278                     not_found_attempts = 0
279 
280         # print result
281         if result.is_valid():
282             r = {"temperature": result.temperature, "humidity": result.humidity}
283         else:
284             print('Failed to get reading. Is the sensor connected? Is the pin number correct?')
285 
286         # clean the gpio
287         GPIO.cleanup()
288         return(r)
```

Kuva 1. DHT-kirjaston päätason koodi muokattu omaksi start_read-funktioksi.

InsertOne-noodin yhteysmäärittelyt



Kuva 1. Noodin toiminnallisuuden ja kokoelman määrittäminen.



Kuva 2. Tietokannan määrittäminen yhteismerkkijonon ja käyttäjätunnuksen avulla.