Erkka Säilynoja

PYTHON IOT OHJELMISTOKEHITYS

Tekniikka
2022

VAASAN AMMATTIKORKEAKOULU
Tietotekniikka

TIIVISTELMÄ

Tekijä	Erkka Säilynoja
Opinnäytetyön nimi	Python IoT Ohjelmistokehitys
Vuosi	2022
Kieli	suomi
Sivumäärä	48
Ohjaaja	Anna-Kaisa Saari

Työn tarkoituksena oli toteuttaa IoT- järjestelmä, jossa lähetetään anturidataa Arduino Nanolta Raspberry Pi:lle Bluetooth Low Energyn yli sekä käsitellä data Python-ohjelmointikieltä hyödyntäen. Pythonilla valmistetaan myös ohjelmistorajapinta, josta WordPress voi lukea anturidataa reaaliaikaisesti ja visualisoida sen.

Työssä perehdytään Bluetooth Low Energy -tekniikkaan ja Python-ohjelmointikieleen kirjastoineen. BLE-yhteyden muodostamiseen käytettynä kirjastona toimi bluepy -kirjasto. Työ eteni tutkimalla internetistä mahdollisia toteutusteknologioita ja yhdistämällä niistä kokonaisuus.

Opinnäytetyön tuloksena voidaan hyödyntää Bluetooth Low Energy- teknologiaa ja Pythonia anturidatan lukemiseen ja viemiseen WordPressille reaaliajassa.

Avainsanat Python, Raspberry Pi, sulautetut järjestelmät, BLE, API, IoT

VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES
Tietotekniikka

ABSTRACT

Author	Erkka Säilynoja
Title	Software development for IoT using Python
Year	2022
Language	Finnish
Pages	48
Name of Supervisor	Anna-Kaisa Saari

The purpose of this thesis was to create an IoT system which consist of sending sensor data from Arduino Nano to Raspberry Pi using Bluetooth Low Energy and handling the data using the Python programming language. Python will be used to make an application programming interface, where WordPress can capture the data real time and visualize it.

In the thesis Bluetooth Low Energy and Python programming language with libraries were examined. The Bluepy library was used to connect to the BLE. The project progressed by researching possible implementation technologies and combining them as a working project.

As a result of this thesis the BLE technology with Python can be utilized to read and push data to WordPress real time.

Keywords Python, Raspberry Pi, embedded systems, BLE, API, IoT

SISÄLLYS

KUVIO- JA TAULUKKOLUETTELO

LYHENNELUETTELO

API	Application programming interface, ohjelmointirajapinta.
BLE	Bluetooth Low Energy, langaton tiedonsiirto protokolla.
GAP	Generic Access Profile, Bluetooth hallinta – ja mainostusprofiili.
GATT	Generic Attribute Profile, tiedonvälitys muoto Bluetooth Low Energy:ssä.
HTTP	Hypertext Transfer Protocol, hypertekstin siirtoprotokolla.
IoT	Internet Of Things, esineiden internet.
I2C	Inter IC, laitteistoläheinen ohjaus- ja tiedonsiirtoväylä.
JSON	JavaScript Object Notation, tiedostomuoto tiedonvälitykseen.
PDM	Pulse-Density modulation, pulssinleveysmodulaatio.
REST	Representational State Transfer, arkkitehtuurimalli ohjelmointirajapintojen toteuttamiseen
RRA	Round Robin Archive, RRDtoolin käyttämä tiedontalletus arkisto.
RRD	Round-robin database, round-robin tietokanta.
RRDtool	Round-robin database tool, round-robin tietokantatyökalu.
TCP	Transmission Control Protocol, tietoliikenneprotokolla tietokoneiden väliseen tiedonsiirtoon.
WSGI	Web Server Gateway Interface, web-palvelinten välinen kommunikointi tapa

1 JOHDANTO

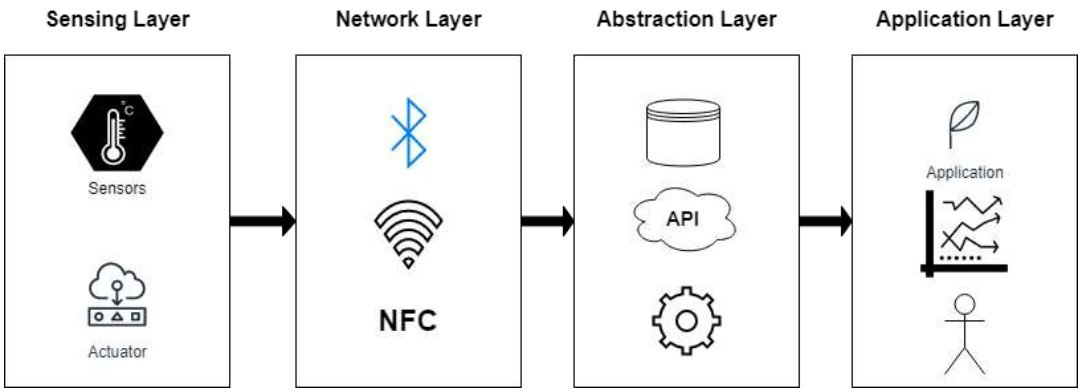
IoT eli esineiden internet kasvaa nopeasti ja yhä useampia laitteita on mahdollista liittää verkkoon. Verkkoon liitetyistä laitteista voidaan kerätä tietoa kuten sensoridataa tai laitteita voidaan ohjata verkon yli. IoT mahdollistaa datan keräämisen reaaliajassa ja sen välittämisen analysoitavaksi. Tätä voidaan hyödyntää esimerkiksi kodin automaatiossa. Kotiin asennetaan antureita seuraamaan lämpötilaa ja kosteutta. Anturin havaitessa lämpötilan liian matalaksi nostetaan lämpötilaa automaattisesti tai ilmankosteuden laskevan liian matalalle käynnistetään ilmankostutin automaattisesti. Mahdollisuudet ovat rajattomat IoT:n hyödyntämisessä teknologiassa. Tässä opinnäytetyössä perehdytään Bluetooth Low Energy -tiedonvälitykseen ja Python-ohjelmistokehitykseen esineiden internetissä.

Arduino Nano 33 BLE Sense:ssä on Bluetooth Low Energy -ominaisuus, jota hyödynnetään opinnäytetyössä datan siirtoon. Työssä luetaan erilaisia antureita Arduino Nano 33 BLE Senseltä ja antureiden arvot lähetetään BLE:n yli Raspberry Pi:lle. Python-ohjelmointi kieltä hyödyntäen muodostetaan yhteys Arduinolle ja vastaanotetaan Arduinon lähettämää dataa. Vastaanotettu data käsitellään ihmiselle luettavaan muotoon ja tarjota WordPressille ohjelmistorajapintaan poimittavaksi. Rajapintoja käytetään yleisesti IoT-maailmassa tiedon välitykseen antureilta verkkoon. Tässä tapauksessa tietoa välitetään rajapinnan kautta WordPressille, jossa anturi dataa esitellään diagrammeilla ja kaavioilla. Opinnäytetyö aiheen tarjoaa Delektre Oy.

1.1 IoT-järjestelmän rakenne

IoT-järjestelmä voidaan jakaa neljään peruskerrokseen. Nämä neljä kerrosta ovat fyysiset laitteet kuten anturit, tiedon välitys, tiedon käsittely ja sovelluskerros, jossa kerätty tieto havainnollistetaan ja käytetään. IoT ei perustu vain yhteen tiet-

tyyn teknologiaan vaan siinä valitaan omaan käyttötarkoitukseen parhaat teknologiat ja ne yhdistetään toimivaksi kokonaisuudeksi. Tästä kokonaisuudesta muodostuu IoT-järjestelmän käsite.



Kuvio 1. IoT-järjestelmän kerroksittainen perusrakenne

Kuviossa 1 havainnollistetaan IoT-järjestelmän kerroksittaista perusrakennetta. Sensing Layer pitää sisällään anturit ja älylaitteet, joita halutaan ohjata tai niiden tietoa halutaan kerätä. Kyseinen kerros voi pitää sisällään useita sensoreita. Näiden laitteiden ohjaus ja tiedon välitys tapahtuu Network Layer:in kautta.

Tiedon välitykseen Network Layer:llä löytyy useita erilaisia teknologioita, kuten matkapuhelinverkkoteknologiat 3G, 4G ja 5G. Nämä teknologiat sopivat hyvin suurten tietomäärien siirtoon. Ongelmaksi näissä teknologioissa kuitenkin muodostuu IoT-järjestelmissä suuri virrankulutus. Mikäli IoT-laite, joka kerää dataa ja lähettää sitä toimii akulla, sille halutaan mahdollisimman suuri akunkesto, joka toimisi mahdollisesti jopa vuosia ilman ylimääräisiä huoltotoimenpiteitä. Opinnäytetyössäkin käytetty BLE-teknologia sopii hyvin IoT-laitteisiin sen pienen energian kulutuksensa takia. BLE:n käytössäkin on kuitenkin haittapuolensa, esimerkiksi lyhyt kantama. Teoriassa BLE:n maksimi kantama voi olla noin 100 metriä, mutta todellisuudessa laitteen optimaalisen toiminnan kannalta kantama on noin 10–20 metriä. Tästä syystä BLE ei sovellu kaikkien IoT-järjestelmien tiedonvälitykseen.

Kolmannessa kerroksessa (Abstraction Layer) tapahtuu datan käsittely, prosessointi sekä mahdollinen talletus tietokantaan. Tämä kerros pitää sisällään muun muassa rajapinnan, joka toimii välikätenä sovelluksen ja antureiden välillä. Rajapinnassa talletetaan anturilukemat tietokantaan ja rajapinnan kautta voidaan hakea tietoa sovellukselle esiteltäväksi ja analysoitavaksi.

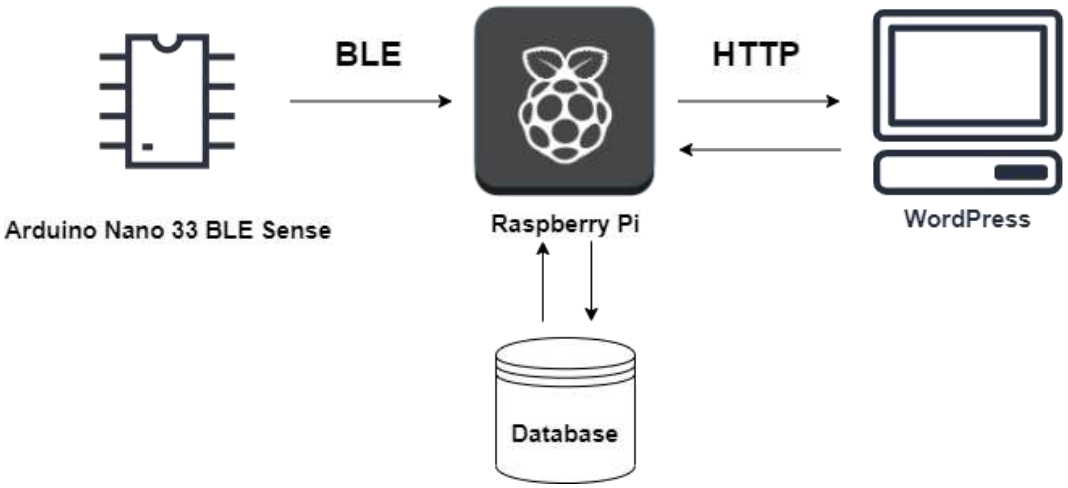
Neljäs kerros on sovelluskerros, jossa tapahtuu tiedon välitys loppukäyttäjälle. Tässä kerroksessa tiedonvälitys tapahtuu tavallisesti HTTP-protokollan mukaisesti. Sovelluskerroksella voi olla useita eri käyttötarkoituksia, kuten hallita IoT-laitteita, tässä tapauksessa antureita. Tässä sovelluksessa ei ollut tarvetta sensoreiden ohjaukselle sovellukselta. Kerroksella voidaan myös analysoida dataa esimerkiksi muodostamalla datasta diagrammeja ja kaavioita. Diagrammien avulla loppukäyttäjä voi seurata antureita ja niiltä saapuvia arvoja. Datasta pyritään kaavioiden avulla etsimään informaatiota, minkä avulla voidaan muodostaa esimerkiksi liiketoiminnan kannalta tärkeitä päätöksiä. Analytiikan lopputarkoitus on siis tarjota dataan perustuen ennusteita, parantaa toiminnan tehokkuutta sekä laatua. (Altexsoft, 2021)

1.2 Delektre Oy

Delektre Oy on vuonna 2010 perustettu vaasalainen yritys, joka on keskittynyt teknologiantutkimus- ja kehittämistehtäviin. Delektre toimii yhteistyössä Euroopan johtavien tutkimusinstituutioiden kanssa. Delektren kehittämiin laitteisiin kuuluu muun muassa Senno Ring, joka on sormus, jonka avulla käyttäjä voi seurata muutoksia kehossaan ja ylläpitää terveellisiä elämäntapoja. (Delektre Oy, 2021)

2 LAITTEISTO JA KÄYTETYT TEKNOLOGIAT

Tässä kappaleessa tutustutaan projektissa käytettyihin laitteistoihin sekä teknologioihin, joita opinnäytetyössä käytettiin.



Kuvio 2. Projektin yleisarkkitehtuuri

Kuviossa 2 havainnollistetaan opinnäytetyön yleisarkkitehtuuria sekä projektissa käytettyjä laitteistoja. Arduinolta lähetetään BLE:n yli anturi lukemat Raspberry Pi:lle, jossa tiedot talletetaan tietokantaan. WordPressin ja tietokannan välinen keskustelu tapahtuu HTTP-protokollan ylitse rajapintaa hyödyntäen. Raspberry Pi:lle on toteutettu rajapinta, joka mahdollistaa Arduinon ja Wordpressin välisen tiedon välityksen.

2.1 Laitteisto

2.1.1 Arduino Nano 33 BLE Sense

Arduino Nano 33 BLE Sense on Arduinon pienikokoinen kehitysalusta, joka sisältää useita antureita. Kehitysalustan koko on vain 45x18 mm. Kehitysalusta sisältää 5 erilaista anturia, jotka mahdollistavat useiden eri mittausten suorittamisen laitteella. Anturit mahdollistavat muun muassa lämpötilan, ilmankosteuden, äänen, valoisuuden sekä liikkeen suunnan ja kiihtyvyyden mittauksen. Laite sopii hyvin

anturitiedon keräämiseen ja välittämiseen. Laitteen mikro-ohjaimena toimii nRF52840, jonka valmistaja on Nordic Semiconductors. Mikro-ohjaimessa on tuki Bluetooth 5.2:lle, joka mahdollistaa datan siirron BLE:n välityksellä. (Arduino.cc, 2021)

Kuvio 3. Arduino Nano 33 BLE Sense

2.1.2 Raspberry Pi

Raspberry Pi on ARM-mikroprosessorilla toimiva yhden piirilevyn tietokone, joka on vain hieman luottokorttia suurempi. Raspberry Pi on tarkoitettu halvaksi ja pieneksi tietokoneeksi lähinnä ohjelmoinnin opetteluun, mutta sen käyttömahdollisuudet ovat lähes rajattomat. Raspberry Pi on suunniteltu käytettäväksi Linuxilla. (Raspberry Pi, 2021)

Työssä Raspberry Pi toimii BLE-vastaanottolaitteena ja Raspberry Pi:llä pyöritetään Flask -rajapintaa, josta voidaan hakea tietoa antureilta.

Kuvio 4. Raspberry Pi 4

2.2 Arduino IDE

Arduino IDE on Arduinon ilmainen ohjelmisto, jolla voidaan kirjoittaa koodia ja siirtää se Arduinon laitteille. Ohjelmistossa on tuki C – ja C++ kielille. Ohjelmistoa käytettiin Arduino Nanon ohjelmoimiseen. (Arduino IDE, 2021)

2.3 Bluetooth Low Energy (BLE)

Opinnäytetyössä Arduino Nano 33 BLE Sense toimii BLE-oheislaitteena, joka kerää dataa ja lähettää datan hyödyntäen BLE-teknologiaa. Tässä kappaleessa perehdytään Bluetooth Low Energy-teknologiaan.

Bluetooth Low Energy (BLE) on langaton teknologia, jota käytetään datan siirtoon lyhyillä kantamilla. BLE-teknologian tarkoituksena on tarjota matala energiankulutus samoilla kantamilla kuin tavallisessa bluetoothissa. Bluetoothin kantamaan vaikuttavat monta tekijää, esimerkiksi ympäristö, jossa Bluetoothia käytetään. Ulkoilmaympäristössä kantama on suurempi kuin sisätiloissa, koska ulkoilmassa ei ole esteitä, kuten laseja ja seiniä. Sekä tavallinen Bluetooth että Bluetooth Low Energy toimivat 2.4GHz taajuudella. Tavallisesti BLE:n yli siirretään pieniä määriä

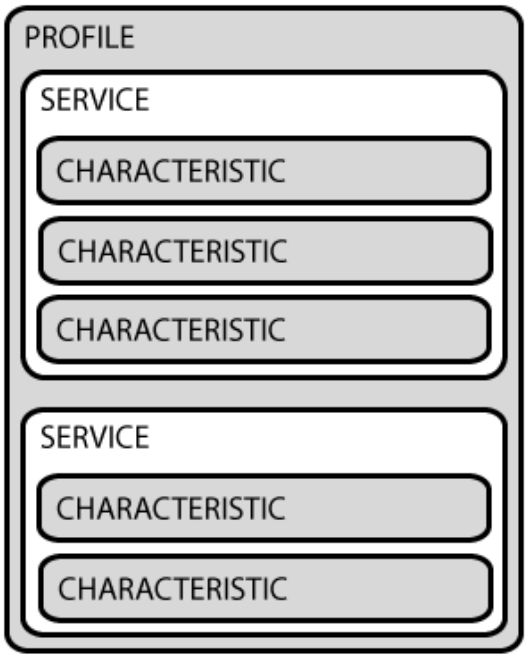
dataa, kuten lämpötilatietoja. Tavallinen bluetooth yhteys sopii paremmin suurien tietomäärien siirtoon. IoT-laitteille usein tärkeä ominaisuus on matala energiankulutus. Pienen energian kulutuksensa vuoksi BLE sopii IoT- laitteisiin. (Bluetooth, 2021)

2.3.1 Generic Access Profile (GAP)

GAP määrittelee tavan, miten laitteet ovat vuorovaikutuksissa toisilleen ja laitteiden roolit toisilleen BLE-tiedonvälityksessä. Toinen laite toimii palvelimena ”Peripheral” laitteena eli oheislaitteena ja toinen laite toimii asiakkaana ”Central” laitteena eli päätelaitteena. Oheislaitteen vastuulla on kerätä data ja mainostaa sitä päätelaitteille. Oheislaitteelle asetetaan aikaväli, jonka välein se lähettää mainostettua pakettia päätelaitteelle. Mitä pidempi aikaväli on, sitä vähemmän laite kuluttaa energiaa. Pienempi virrankulutus perustuu siihen, että aikavälin väliajan radio on pois päältä ja dataa ei välitetä. Opinnäytetyössä Arduino Nano 33 BLE toimii oheislaitteena ja Raspberry Pi päätelaitteena. (Adafruit.com, 2021)

2.3.2 Generic Attribute Profile (GATT)

GATT määrittelee tavan, jolla kaksi BLE-laitetta välittävät dataa keskenään. BLE-laitteet välittävät dataa toisilleen käyttäen konsepteja palvelut (Services) ja ominaisuudet (Characteristics). Tämä on eräänlainen taulu, jonka uloin kerros on profiili ja toinen kerros on palvelut. Palveluiden sisällä on ominaisuudet, joihin lähetettävät arvot talletetaan. BLE-laitteella voi olla useampi palvelu, jotta data voidaan jakaa sopiviin lohkoihin. Yksi palvelu voi sisältää monta ominaisuutta, jotka sisältävät datan. Ominaisuuksiin voidaan esimerkiksi tallettaa lämpötilatieto ja sitten mainostaa palvelua, joka sisältää kyseisen ominaisuuden. Näin lähetetty data jaetaan sopiviin lohkoihin. Attribuutin arvon on mahduttava yhteen pakettiin ja datapakettien koko on rajoitettu. Bluetooth 4.0 ja 4.1 data paketin koko on 20 tavua ja 4.2 sekä 5.0 paketin koko on rajoitettu 244 tavuun. (O’Reilly, 2021)



Kuvio 5. GATT-rakenne

Kuviossa 5 havainnollistetaan GATT-rakennetta. Jokaisen palvelun sisällä on ominaisuudet, joihin talletetaan anturilukemat.

2.4 Python

Python on monipuolinen ohjelmointikieli, joka sopii monenlaisiin tehtäviin, kuten skripti- kieleksi, jolla voidaan suorittaa esimerkiksi automaatioita palvelinympäristöissä. Python sopii erinomaisesti myös web-kehitykseen. Web-kehitykseen sopivia web-kehyksiä ovat muun muassa Django ja Flask. Muita käyttötarkoituksia Pythonilla on tieteellinen laskenta, datatieteet, koneoppiminen sekä ohjelmoinnin opetus. Python soveltuu sekä oliopohjaiseksi että funktionaaliseksi ohjelmointikieleksi. Tämä tekee Python-kielestä erittäin skaalautuvan moniin eri käyttötarkoituksiin.

Python on avoimen lähdekoodin ohjelmointikieli ja toimii lähes kaikilla alustoilla. Python on tulkattava kieli eli lähdekoodia ei käännetä konekieliseksi koodiksi, vaan ohjelma suoritetaan Python-tulkin avulla. Tulkattavuus kuitenkin tekee Pythonista

hieman hitaamman kuin perinteiset käännettävillä kielillä kirjoitetut ohjelmat. Tällainen käännettävä kieli on esimerkiksi C-kieli, jolla tässä opinnäytetyössä kirjoitettiin Arduino Nanolle suoritettava ohjelmisto. Tosin Pythoniin on saatavilla C-laajennus, mikäli ohjelmalta vaaditaan tehokkaampaa suorituskykyä. Tämä laajennus on nimeltään Cython. Tätä laajennusta ei käytetty opinnäytetyössä, koska rajataan alue, johon perehdytään opinnäytetyössä.

Python-ohjelmointikieli jakautuu versioihin 2 ja 3, jotka eivät ole keskenään yhteensopivia. Ohjelmissa suositellaankin käytettävän nykyään Python 3 versiota, koska version 2 kehittäminen on päättynyt vuonna 2020 ja siihen ei ole enää saatavilla tietoturvapäivityksiä. (Wikipedia Python, 2021)

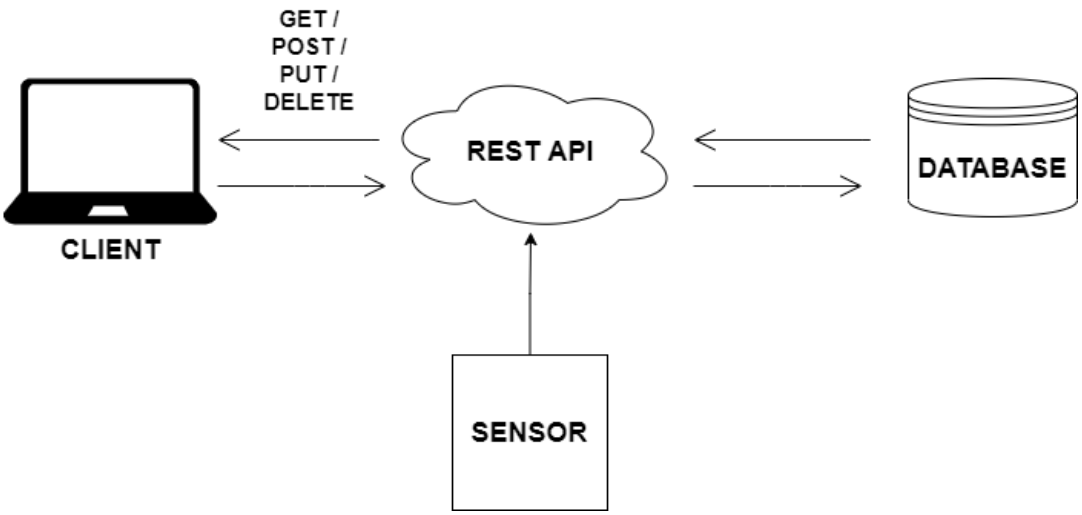
2.5 Ohjelmointirajapinta

Ohjelmointirajapinta (API) mahdollistaa eri sovellusten välisen kommunikaation. Rajapintoja käytetään kahden erillisen ohjelmiston väliseen ohjelmalliseen yhdistämiseen. Rajapinnan käyttö mahdollistaa sovellusten välisen kommunikaation ja tiedonvälityksen.

Tässä projektissa kehitetyn API:n avulla voidaan tehdä pyyntöjä ja kutsuja palvelimelle tiedonkeruutarkoituksessa. Kommunikointi tapahtuu JSON-muodossa, joka on yksinkertainen tiedostomuoto ohjelmistojen väliseen keskusteluun. Rajapinnan avulla tietoa voidaan myös muuttaa tai poistaa. Toiminta perustuu HTTP-protokollaan. Http-protokolla toimii niin, että asiakasohjelma avaa palvelimelle TCP-yhteyden ja lähettää pyynnön. Rajapinta vastaa pyyntöön lähettämällä JSON-vastauksen. Yleisimmät metodit keskustella rajapinnan kanssa ovat GET, PUT, POST ja DELETE. GET-pyynnön avulla haetaan tietoa rajapinnalta. Asiakasohjelma voi esimerkiksi pyytää GET-metodilla palvelimelta lämpötilatietoja viimeisimmän tunnin ajalta. Palvelin palauttaa JSON-muodossa viimeisen tunnin sensorilukemat. POST-metodilla lisätään tietoa eli saman esimerkin mukaisesti voidaan lisätä uusi sensori

tai lähettää uutta tietoa jo valmiiksi tunnetulta sensorilta ja tallettaa tiedot tietokantaan. DELETE-metodilla voidaan nimensä mukaisesti poistaa tietoja, kuten voidaan poistaa tietokannalta anturi, joka ei ole enää käytössä. GET-pyynnössä data kirjoitetaan URL-osoitteeseen, kun taas POST-pyynnössä data lähetetään HTTP-pyynnön body- osiossa. GET-pyynnössä voidaan asettaa argumentteja URL-osoitteeseen ja hakea tällä tavoin dataa palvelimelta. (Wikipedia Ohjelmointirajapinta, 2021)

Rajapinnalle määritellään päätepisteet, jotka ovat URL-osoitteita, joista dataa voi hakea, lisätä, poistaa tai muokata. Rajapinnan pääteosoite voisi olla esimerkiksi http://esimerkki/sensors. Kyseisen pääte osoitteen kautta voitaisiin tehdä POST- ja GET-pyyntöjä. Tekemällä GET-pyyntö osoitteeseen saadaan lista antureista JSON-muodossa, joita rajapinnan kautta voidaan lukea. Jokaisella anturilla on numeerinen id eli tunniste, jonka avulla voidaan suunnata jatko-osoitteeseen, josta saadaan lisätietoja anturilta ja anturin lukemat. Osoite tälle voisi olla http://esimerkki/sensors/{id}. Kaarisulkeiden sisälle sijoitetaan anturin id, kyseinen id on muuttuja, joka toimii tunnisteena anturille.



Kuvio 6. Projektin REST API -arkkitehtuuri

Kuviossa 6 esitellään projektin REST API -arkkitehtuuria. Kuviossa anturilta lähetetään POST-metodilla dataa rajapinnalle, jossa se talletetaan tietokantaan. Rajapinnan kautta tietokannassa oleva data on saatavilla asiakassovellukselle. REST on HTTP-protokollaan perustuva arkkitehtuurimalli ohjelmointirajapintojen toteuttamiseen. Sen päällimmäisenä tarkoituksena on parantaa rajapintatoteutusten suorituskykyä, skaalautuvuutta, yksinkertaisuutta sekä luotettavuutta. (Wikipedia REST, 2021)

2.6 JSON

JSON (JavaScipt Object Notation) on kevyt kieliriippumaton tiedostomuoto tiedonvälitykseen. Nimestään huolimatta JSON on JavaScriptistä riippumaton. JSON on ihmisille helposti ymmärrettävä tiedostomuoto ja koneille helposti jäsenneltävissä ja muodostettavissa. JSON:ia käytetään tavallisesti datan säilömiseen ja välittämiseen. REST API arkkitehtuureissa palvelimen toiminnot tapahtuvat JSON muodossa. JSON koostuu kokoelmista nimiä ja arvoja. Arvot voidaan esittää JSON-tiedostossa listana tai objektina. JSON:ssa on tuki 6 erilaiseen tietotyyppiin: merkkijono (string), numerot, totuusarvomuuttuja (boolean), null, objekti sekä lista. JSON-listamuotoinen olio esitetään seuraavasti:

```
[{"id":1,"name":"temp1","s_type":"temperature"},
{"id":2,"name":"pressure1","s_type":"pressure"},
{"id":3,"name":"humidity1","s_type":"humidity"},
{"id":4,"name":"luminosity1","s_type":"luminosity"},
{"id":5,"name":"mic1","s_type":"sound"},
{"id":6,"name":"accelometer1","s_type":"accelometer"}]
```

Esimerkissä “id” toimii avaimena ja ensimmäisen id:n arvo on yksi. Avain erotetaan sen arvosta kaksoispisteen avulla. JSON syntaksissa hakasulkeet muodostavat listan(array) ja kaarisulkeet muodostavat olion (object). Olion sisällä data esitetään avaimin ja arvoin, nämä erotellaan toisistaan kaksoispisteen avulla. (IETF, 2017)

2.7 Flask

Flask on kevyt pythonilla kirjoitettu WSGI (Web Server Gateway Interface) web kehys. Flask soveltuu hyvin rajapintojen kehitykseen, koska se on suunniteltu helposti lähestyttäväksi. Flask skaalautuu hyvin myös laajempiin toteutuksiin. Flaskiin on tarjolla laajat kirjastot, jotka mahdollistavan monipuolisenkin applikaation toteutuksen. Tunnettuja Flaskilla toteutettuja sovelluksia ovat muun muassa Pinterest ja LinkedIn. (Full Stack Python, 2021)

Flask-RESTful on Flaskin laajennusosa ja sen avulla voidaan rakentaa nopeasti ja helposti REST API. Flask-RESTful on kevyt laajennus, joka toimii muidenkin Flask-kirjastojen kanssa. Kirjaston avulla voidaan luoda yhteen URL-osoitteeseen monta HTTP-metodia, kuten POST, GET ja DELETE. Tätä tapaa luoda rajapinta käyttäen Flask-RESTful -laajennusta kutsutaan termillä ”Resourceful Routing”. Yksikertaisimmillaan rajapinta luodaan käyttäen tätä kirjastoa kuvion 7 mukaisesti.

```
from flask import Flask
from flask_restful import Resource, Api

app = Flask(__name__)
api = Api(app)

class HelloWorld(Resource):
    def get(self):
        return {'hello': 'world'}

api.add_resource(HelloWorld, '/')

if __name__ == '__main__':
    app.run(debug=True)
```

Kuvio 7. Resourceful Routing esimerkki 'Hello world' -toteutuksesta

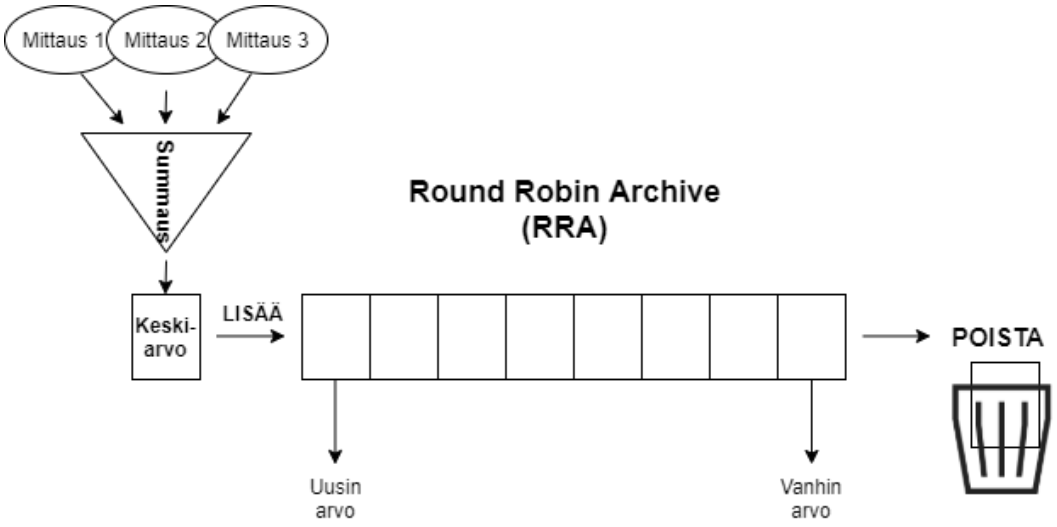
Kuviossa 7 luodaan yksinkertainen API käyttäen Flask-RESTful laajennusta. Esimerkissä luodaan luokka HelloWorld, jossa on HTTP-metodi GET, joka palauttaa kutsuttaessa vastauksena ” {"hello": "world"}”. Samaan luokkaan voitaisiin lisätä tarvittaessa POST, DELETE ja PUT metodit. (Flask-Restful, 2021)

2.8 RRDTool

RRDtool (Round Robin Database -työkalu) on avoimen lähdekoodin työkalu, jolla käsitellään aikasarjallista dataa, kuten esimerkiksi lämpötilatietoja. RRDtoolia voidaan käyttää muun muassa shell-skripteissä, perlissä, pythonissa ja rubyssa. RRDtoolilla luodaan Round Robin tietokantoja, joista voidaan myös muodostaa kuvaajia. Tavallisessa lineaarisessa tietokannassa tietokannan pohjalle lisätään jatkuvasti uutta dataa ja tietokannan koko kasvaa jatkuvasti datanmäärän lisääntyessä. Round Robin tietokannassa tietokannan koko määritellään sen luontivaiheessa. Round Robin tietokantaa voidaan ajatella ympyränä, johon talletetaan arvoja. Kun ympyrän lähtöpiste on saavutettu, tietoa aletaan tallettamaan vanhojen arvojen päälle, jolloin tietokannan koko pysyy muuttumattomana.

Tietokannassa asetetaan aikaväli (step), jonka välein tietoa kerätään ja tieto talletetaan RRA:han (Round Robin Archive). RRA on parametri tietokannassa, jonne

tietoa talletetaan. Aikaväli voi olla esimerkiksi 10 sekuntia. RRA:han voidaan tallettaa tietoa jokaisen aikavälin kohdalla tai vaihtoehtoisesti voidaan muodostaa keskiarvo esimerkiksi kuudesta askeleesta eli tietoa talletetaan 60 sekunnin välein. Keskiarvojen tallettaminen säästää tilaa sekä vähentää virhelukemia, joita voidaan saada sensoreilta, joiden tietoa halutaan säilöä.



Kuvio 8. Round Robin -tietokannan periaatteet

Kuviossa 8 havainnollistetaan, kuinka tietoa talletetaan RRA:han. Jos tietokantaan halutaan säilöä dataa vuorokauden ajalta askelvälin ollessa 10 minuuttia ja tietokantaan halutaan säilöä keskiarvo jokaiselta tunnilta otetaan mittaukset 10 minuutin välein tunnin ajalta ja näistä kuudesta mittauksesta muodostetaan keskiarvo ja tämä säilötään RRA:han. Samaan aikaan tietokannasta poistetaan vuorokausi sitten talletettu tunnin keskiarvo. Jos mittauksia ei ole tunnin aikana tullut, talletetaan se tunnin kohdalle arvo ”null”. (Bogaerdt A, 2019)

2.9 SQLite

SQLite on kevyt tietokantaohjelmisto, joka on toteutettu C-kirjastona. SQLite on SQL-pohjainen tietokanta. Tavallisesta relaatiotietokannasta poiketen tietokanta

linkitetään sitä käyttävään sovellukseen. Tästä syystä SQLite ei vaadi erillistä tietokantapalvelinta tai tietokannanhallintaohjelmistoa. Tietokannan tallennustilana toimii tietokoneen muisti. Näistä syistä johtuen SQLite on erinomainen vaihtoehto sovellusohjelmiin. Opinnäytetyössä SQLite toimii tietokantana, jonne talletetaan sensoreilta saapuva data. Kappaleessa 3.5 perehdytään lisää tietokantoihin. (Wikipedia SQLite, 2021)

3 SUUNNITTELU

Tässä luvussa perehdytään projektin vaatimuksiin sekä tutkimustyöhön, jonka perusteella projekti toteutettiin.

3.1 Vaatimusmäärittely

Taulukossa 1 kuvataan vaatimukset, joiden perusteella ohjelmaa lähdettiin toteuttamaan.

Taulukko 1. Vaatimusmäärittely

Nro	Kuvaus	Tärkeys
1	Lue sensoreita Arduinolta	1
2	Lähetä sensori lukemat 5 sekunnin välein käyttäen BLE	1
3	Vastaanota Bluetooth Low Energy:n välityksellä saapuvat paketit	1
4	Käännä paketit ihmiselle luettavaan muotoon	1
5	Luo ohjelmistorajapinta	1
6	Lähetä data rajapintaan	1
7	Tarjoa rajapinnasta dataa WordPressille luettavaksi	1
8	Talleta sensorit ja sensorilukemat tietokantaan	1
9	Hae dataa tietokannasta halutun sensorin ID:llä	1
10	Hae dataa tietokannasta ajan perusteella	2
11	Lisää uusia sensoreita tietokantaan rajapinnan kautta	3

Taulukossa 1 Tärkeys-sarakkeessa kuvataan vaatimuksen tärkeysastetta. Tärkeysjärjestyksessä 1 kuvastaa tärkeintä ja 3 vähemmän tärkeää tehtävää. Ohjelmisto toteutettiin vaatimusmäärittelyn tärkeysjärjestyksen mukaan. Ensin toteutetaan tärkeimmät vaatimukset ja ohjelmistoa jatketaan vähemmän tärkeillä vaatimuksilla.

3.2 Vaadittavat kirjastot Arduinolla

Arduino Nano 33 BLE SENSE sisältää useita antureita. Antureiden hyödyntämistä ja lukemista varten täytyy ohjelmakoodiin tuoda jokaiselle anturille oma kirjastonsa, joka sisältää luokat ja funktiot anturin käsittelyyn. Seuraavissa osioissa käydään läpi tarvittavat kirjastot antureiden lukemiseen ja BLE-ominaisuuden käyttöönottoon.

3.2.1 ArduinoBLE

ArduinoBLE on bluetooth kirjasto, joka toimii kaikissa BLE-ominaisuuden omaavissa Arduino laitteissa. Kirjastossa on tuki sekä tavalliseen Bluetoothiin että Bluetooth Low Energyyn. Opinnäytetyössä käytetään Bluetooth LE:tä sen vähäisen virran kulutuksen takia. Vähäinen virrankulutus saavutetaan pelkistetyllä liikennöinnillä BLE:n yli. Kirjasto toimii GATT-protokollan mukaisesti. GATT-protokolla määrittelee kommunikaatiotavan Bluetooth Low Energy -laitteiden välillä. GATT-protokollan mukaisesti toinen laite toimii GATT palvelimena ja toinen GATT-asiakkaana. Tässä tapauksessa Arduino Nano toimii ”Peripheral” laitteena eli GATT-palvelimena.

Kirjastossa on lukuisia eri luokkia erilaisilla toiminnallisuuksilla. BLE-luokkaa käytetään BLE-moduulin käyttöönottoon. *BLEDevice* -luokkaa käytetään saadakseen tietoa liitetyistä laitteista. *BLEService* -luokalla otetaan ominaisarvoja ”Characteristic” sisältävät palvelut käyttöön. *BLECharacteristic* -luokalla voidaan lukea ja kirjoittaa arvoja/muuttujia, joita halutaan BLE:n yli lähettää. (arduino.cc, 2021)

3.2.2 Arduino_HTS221

ArduinHTS221 kirjaston avulla voidaan käyttää Arduino Nano 33 BLE Sensen HTS221 sensoria ja lukea anturilta lämpötila ja kosteustiedot. Kirjasto antaa kosteuslukeman väliltä 0-100 % ja lämpötilan puolen asteen tarkkuudella -40-120 °C väliltä.

Kirjasto lisätään ohjelmistoon komennolla *#include <Arduino_HTS221.h>*. Kirjasto tarjoaa funktiot *begin()* ja *end()*, joilla anturi alustetaan ja suljetaan. Lämpötilan ja kosteuden lukemiseen kirjastossa on funktiot *readTemperature()* ja *readHumidity()*. (arduino.cc, 2021)

3.2.3 Arduino_LPS22HB

ArduinoLPS22HB mahdollistaa LPS22HB anturin käytön ja ilmanpaineen lukemiseen anturilta. Kirjasto täytyy lisätä ohjelmaan, jotta sen funktioita voidaan käyttää. Anturi täytyy alustaa *begin()* funktiolla, jotta voidaan lukea ilmanpaine anturilta. (arduino.cc, 2021)

3.2.4 Arduino_APDS9960

ArduinoAPDS9960-kirjasto mahdollistaa APDS9960-anturin käytön Arduino Nanolla. Kirjastoa käyttäen sensorilta voidaan lukea asentoa, väriä, valon intensiteettiä sekä etäisyyttä. Sensorilla voidaan seurata käden asentoa käyttäen 4:ää fotodiodia, jotka sijaitsevat sensorin sisällä. Etäisyyttä mitataan sillä, kuinka paljon infrapunavaloa heijastuu takaisin sensorin edessä olevasta esineestä. Kirjastoa käytetään importoimalla se ohjelmakoodiin. (arduino.cc, 2021)

3.2.5 Arduino_LSM9DS1

Arduino_LSM9DS1 kirjasto mahdollistaa Arduino Nano 33 BLE:n IMU anturin käytön. Anturia käytetään havaitsemaan kiihtyvyyttä ja asentoa (gyroskooppi). Anturilla on myös magnetometri, mutta sitä ei käytetä opinnäytetyössä. LSM9DS1 sensori on yhdistetty Arduino Nano mikro-ohjaimeen I2C väylän kautta. Kiihtyvyysmittarin mitattava väli on anturilla [-4, +4] g ja sen tarkkuus on -/+0,122 mg. Gyroskoopin mitattava alue on [-2000, +2000] dps +/-70mdps. (arduino.cc, 2021)

3.2.6 PDM

Tavallisesti mikrofonilla mitataan analogista signaalia (ääni), joka muunnetaan digitaaliseksi A/D muunnoksella. PDM eli pulssileveysmodulaatio on modulaation muoto, jossa analoginen signaali esitetään binäärisenä signaalina. PDM kirjasto mahdollistaa pulssitiheyden modulaation mikrofoneilla, kuten Arduino Nano 33 BLE Sense:n MP34DT05 sensorin lukemisen. Pulssitiheys kertoo mitattavan analogisen signaalin amplitudin eli äänen voimakkuuden. (arduino.cc, 2021)

3.3 Python BLE-kommunikaatio

Pythonissa on muutamia mahdollisia kirjastoja Bluetooth LE datan vastaanottamiseen. Tässä osiossa perehdytään lähinnä bluepy -kirjastoon, jota päädyttiin käyttämään tässä opinnäytetyössä.

Kirjaston avulla tulisi olla mahdollista muodostaa yhteys Arduino:lle käyttäen BLE:tä sekä vastaanottaa BLE:n ylitse lähettyjä anturilukemia. Kirjasto toimii Linux-käyttöjärjestelmillä ja se on testattu käyttäen Python 3:sta. Kirjasto toimii BlueZ pohjalta, joka on Linux-käyttöjärjestelmien virallinen Bluetooth-pino ja tarjoaa tuen Bluetooth protokollille Linux-käyttöjärjestelmissä. Tästä syystä bluepy -kirjasto on oivallinen vaihtoehto käytettäväksi Raspberry Pi:lle sen käyttöjärjestelmän ollessa Linux-pohjainen. (Harvey I, 2014)

Muita mahdollisia kirjastoja BLE-kommunikaation Pythonilla ovat bleak ja pygatt. Molemmat kirjastot tarjoavat samat ominaisuudet sekä toimivat Linux-käyttöjärjestelmän kanssa. Opinnäytetyössä päädyttiin kuitenkin käyttämään bluepy- kirjastoa, sen syntaksin ollessa yksinkertainen ja koska bluepy-kirjaston dokumentaatio on kattavaa.

3.4 Ohjelmistorajapinnan suunnittelu

IoT-laitteen ja käyttöliittymän väliseen kommunikaation tarvitaan rajapinta, jonka avulla voidaan hakea anturilukemat nopeasti ja reaaliajassa. Rajapinta eli API on tarpeellinen silloin kun data on nopeasti muuttuvaa kuten tässä tapauksessa anturidata. Antureilta tulee jatkuvasti uudet lukemat, jotka täytyy välittää rajapinnalle, josta ne voidaan poimia edelleen. Rajapinta toimii palvelimena, jota käytetään useimmiten tiedon välitykseen tai lähettämiseen ohjelmalta toiselle.

Rajapinnan tärkeimmät ominaisuudet ovat anturidatan vastaanotto sekä datan tarjoaminen URL-osoitteen perusteella WordPressille. Rajapinnan muita ominaisuuksia ovat uusien antureiden poisto sekä lisääminen. Tämä lisää rajapinnan skaalautuvuutta sekä jatkokäyttömahdollisuuksia suurempiin järjestelmiin.

Taulukko 2. Rajapinnan toiminnallisuudet.

HTTP-metodi	**URL**	**Kuvaus**
GET	/sensors	Haetaan kaikkien antureiden tiedot.
GET	/sensors/[id]	Haetaan dataa tietyltä anturilta tietokannasta.
POST	/sensors	Lisätään uusi anturi tietokantaan.
POST	/sensors/[id]	Lisätään tietokantaan id:n perusteella uusi anturilukema.
DELETE	/sensors/[id]	Poistetaan anturi.

Taulukon 2 mukaisesti rajapinnan kautta halutaan hakea GET-pyynnöllä aktiiviset anturit sekä halutun anturin lukemat. POST-pyynnöllä voidaan lisätä uusi anturi tai

lähettää jo tunnetun anturin mittaustuloksia tietokantaan talletettavaksi ja tarjottavaksi eteenpäin GET-pyynnöllä. DELETE-metodin avulla voidaan poistaa sensori tietokannasta.

Tärkeänä ominaisuutena rajapinnalle toteutettiin myös mahdollisuus hakea anturidataa rajapinnalta aikaleiman perusteella. Tämä mahdollistaa historiallisen datan hakemisen, jonka pituus voisi vaihdella tunneista vuorokausiin.

Rajapinnan toteutukseen valikoitui Flask sen suorituskyvyn ja skaalautuvuuden vuoksi. Flaskiin on saatavilla laajat kirjastot, jotka mahdollistavat tehokkaan ja yksinkertaisen rajapintatoteutuksen luomisen.

3.5 Tietokanta

IoT-järjestelmään tietokannan valinnassa tulee ottaa huomioon useita tekijöitä. Näitä ovat esimerkiksi päätökset halutaanko tietoa säilöä loputtomiin, miten tietokannan tulisi skaalautua, säilötäänkö tieto pilveen vai paikallisesti. Tietokantaa valittaessa tulee myös ottaa huomioon, kuinka paljon dataa on, mihin sitä käytetään, kuinka monta laitetta on liitetty järjestelmään, sekä montako laitetta järjestelmä tulee pitämään sisällään tulevaisuudessa. Opinnäytetyön lähtökohtana on nykyhetken tilanteen tarkastelu, eli tietokantaan tulee pystyä säilömään Arduino Nanolta saapuvien anturilukemien tiedot.

Jos sensorilaitteelle halutaan mahdollisimman suuri akunkesto, halutaan laitteella minimoida tietoliikennettä. Käytännössä tämä tarkoittaa sitä, että tietoa välitetään esimerkiksi vain viiden sekunnin välein, jolloin virrankulutus on pienempää. Tällöin sopiva tietokanta voisi olla RRDTool. RRDtoolilla voidaan anturilukemien talletus tehdä paikallisesti, silloin kun, radioliikennettä ei ole, joka minimoi laitteen akunkulutusta. RRDtoolin parhaisiin puoliin lukeutuu myöskin sen tietokantojen pieni koko. Kyseisellä työkalulla voidaan tehdä anturilukemien yhteenlaskua ja muodostamaan näistä keskiarvoja, jotka talletetaan tietokantaan. Tallettamalla

tietokantaan keskiarvoja tietokannan koko pidetään minimaalisena. RRD tietokantaan säilötään tietoa aikasarjassa, joten jokaisella talletushetkellä on oma aikaleimansa. Tämä on tärkeää tietoa tarkastellessa sensoridataa. Aikaleiman perusteella voidaan muun muassa havaita, millä ajanhetkellä lukemat ovat muuttuneet ja voidaan tehdä johtopäätöksiä ja selvittää, miksi ne ovat muuttuneet ja mitä tästä on aiheutunut. RRDToolilla voidaan myöskin muodostaa helposti ja nopeasti datasta kaavio, joka ei kuitenkaan ole reaaliajassa päivittyvä kaavio.

SQL-tietokanta on joustava, mutta se vaatii enemmän resursseja. SQL-tietokannassa sensoridata talletetaan taulujen sarakkeisiin ja riveille. SQL vaatii siis selvän struktuurin ja lokerot datalle, mitä sinne talletetaan. SQL-tietokantojen parhaisiin ominaisuuksiin lukeutuu yksinkertaisuus ja mahdollisuus suorittaa tehokkaita hakuja tietokannasta. Suunnitellessa SQL-tietokantaa tulee ottaa huomioon, että tietoa ei talleteta automaattisesti aikasarjassa, vaan sinne tulee luoda oma sarake aikaleimalle, jotta voidaan suorittaa sensoridatalle aikaan perustuvia hakuja.

Toteutettaessa IoT-järjestelmää on SQL-tietokanta usein riittävä vaihtoehto, etenkin pienemmissä järjestelmissä. Nykyään kuitenkin NoSQL-tietokannat ovat kasvattamassa suosiotaan niiden hyvän skaalautuvuuden ja joustavuuden vuoksi. NoSQL-tietokantaan voi tallettaa hyvin sekalaista dataa, esimerkiksi anturidataa, valokuvia, videoita ja äänitiedostoja. NoSQL-tietokannan rakenne ei ole niin tarkkaan määritelty kuin SQL-tietokannan, mikä helpottaa muutosten tekemistä tietokannassa. Tiedon haku NoSQL-tietokannasta on hitaampaa sen rakenteen puutteen vuoksi, ellei suunniteltaessa tietovarastoa ole tätä otettu huomioon.

Opinnäytetyössä tietokannaksi valittiin SQLite sen ollessa pieni, tehokas ja luotettava tietokanta. SQLite on SQL-pohjainen tietokanta, joten siinä hyödynnetään SQL:n parhaita puolia, kuten aiemmin mainittu joustavuus ja tehokkaat hakutoiminnot. SQLite on tiedostopohjainen, joten se ei tarvitse erillistä palvelinta, kuten SQL-tietokannat tavallisesti tarvitsevat. Opinnäytetyössä rajapinta toteutettiin

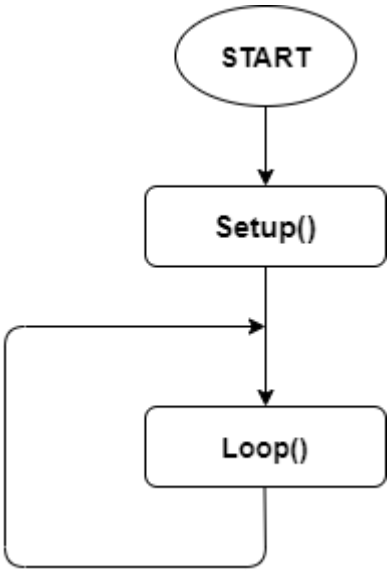
myöskin käyttäen RRDtoolia tietokantana. RRDtool ei kuitenkaan ole yhtä skaalautuva vaihtoehto ja se on myös monimutkaisempi toteuttaa rajapintaympäristöissä. RRDtool soveltuu ominaisuuksiltaan paremmin käytettäväksi suoraan anturilaitteelta, eikä reaaliaikaiseen rajapintaratkaisuun. (Collin J & Saarelainen A, 2016)

4 TOTEUTUS

Tässä kappaleessa tutustutaan opinnäytetyön toteutukseen ja ohjelmistoratkaisuihin.

4.1 Arduino ohjelmointi

Työn ensimmäisessä vaiheessa ohjelmoitiin Arduino Nano 33 BLE Sense. Arduinolta luetaan lämpötilaa, kosteutta, ilmanpainetta, äänenpainetta, kiihtyvyyttä, suuntaa sekä valaistusta. Antureilta saadut lukemat tulee välittää BLE:tä käyttäen Raspberry Pi:lle. Toteutusta varten Arduinolle tulee ladata jokaiselle sensorille oma kirjastonsa sekä BLE-kirjasto. Ohjelman perus rakenteeseen kuuluu *setup()* ja *loop()*. Setup rakenteessa tehdään määrittelyt kuten anturien ja Bluetooth yhteyden alustus.



Kuvio 9. Yleinen Arduino -ohjelmiston virtauskaavio

Kuviossa 9 havainnollistetaan Arduinon virtauskaaviota. *Setup()* -osio toistetaan kerran ja siirrytään *loop()* -osuuteen. *loop()* -osio on niin sanottu ikuinen loop, jota toistetaan, kunnes ohjelma sammutetaan.

4.1.1 Mittaukset

Ohjelmaan täytyy sisällyttää luvussa 3 mainitut kirjastot antureiden lukemista varten. Anturit täytyy aluksi alustaa ohjelmankoodin *setup()* -osiossa. HTS221 ja LPS22HB anturit alustetaan seuraavasti:

```
Serial.println("Initializing humidity and temperature sensor.");
if (!HTS.begin()) {
  Serial.println("Failed.");
  error_pulse();
}

Serial.println("Initializing barometric pressure sensor.");
if (!BARO.begin()) {
  Serial.println("Failed.");
  error_pulse();
}
```

Kuvio 10. Antureiden alustus

Antureiden alustuksen jälkeen voidaan ohjelman *loop()* -osiossa lukea antureilta lukemat ja tallettaa lukema muuttujaan, joka tullaan myöhemmin välittämään BLE:n ylitse. Kuviossa 11, luetaan lämpötilaa ja ilmanpainetta.

```
// Get readings from sensors and update the charcte
pressure = BARO.readPressure(KILOPASCAL);
temperature = HTS.readTemperature(CELSIUS) - 4; //
```

Kuvio 11. Antureiden lukeminen

Funktion *readPressure()* tai *readTemperature()* sisällä voidaan määrittää yksikkö, jossa kyseinen arvo esiintyy.

4.1.2 BLE-käyttöönotto

Arduinolla käytetään kirjastoa ArduinoBLE.h BLE-ominaisuuksien käyttämiseen ja tiedon välittämiseen BLE:n yli. Kirjasto sisällytetään ohjelmaan, jotta sitä voidaan käyttää. BLE:n käyttöön ottamiseksi täytyy ensimmäiseksi luoda palvelu (service)

ja tuntomerkit(characteristic), joihin myöhemmin kirjoitetaan antureiden arvot mainostettavaksi. BLE:n käyttöönotto on esitetty kuviossa 12.

```
// Declare Bluetooth service name, and characteristics.
BLEService environmentalSensingService("181A");
BLEUnsignedIntCharacteristic pressureCharacteristic("2A6D", BLERead | BLENotify);
BLEShortCharacteristic temperatureCharacteristic("2A6E", BLERead | BLENotify); //
BLEUnsignedIntCharacteristic humidityCharacteristic("2A6F", BLERead | BLENotify);
```

Kuvio 12. BLEService ja BLECharacteristic

BLEService -funktiolle annetaan parametrina UUID. UUID annetaan merkkijonona, joka toimii tunnisteena kyseiselle palvelulle. Jokaiselle sensorille täytyy luoda oma tuntomerkki eli ”characteristic” käyttämällä funktiota *BLECharacteristic()*. Kyseisen funktion syntaksi on *BLECharacteristic(uuid, properties, value, valueSize)*. Myöskin tässä funktiossa UUID toimii tunnisteena tietylle tuntomerkille palvelussa. Tuntomerkeille käytetään ominaisuuksia BLERead ja BLENotify, joiden avulla voidaan lukea ja saada ilmoituksia kyseiseltä tuntomerkiltä, jos sen arvo muuttuu. Ohjelman *setup()* -osiossa tulee alustaa Bluetooth Low Energy -tiedonvälitys.

```
// Set up Bluetooth Environmental Sensing service.
Serial.println("Setting up service with characteristics
BLE.setLocalName("Nano33BLE");
BLE.setAdvertisedService(environmentalSensingService);
```

Kuvio 13. Asetetaan BLE-nimi

Kuviossa 13 asetetaan BLE-laitteelle laitteen mainostuksessa käytettävä nimi ja palvelu, johon asetetaan antureiden lukemat. Tämän jälkeen halutut tuntomerkit lisätään haluttuun palveluun. Tuntomerkkien lisääminen tapahtuu kuvion 14 esittämällä tavalla:

```
// Add characteristics for barometric pressure, temperature, and humidity.
environmentalSensingService.addCharacteristic(pressureCharacteristic);
environmentalSensingService.addCharacteristic(temperatureCharacteristic);
environmentalSensingService.addCharacteristic(humidityCharacteristic);
```

Kuvio 14. Tuntomerkkien lisääminen BLE-palveluun

Mainostettavat palvelut lisätätään BLE-yhteyteen kuvion 14 mukaisesti. BLE-yhteydestä tehdään yhdistettävä komennolla *BLE.setConnectable(true)*. Komennon avulla voidaan Arduinoon muodostaa yhteys. Palvelua mainostetaan funktiolla *BLE.advertise()* kuvion 15 mukaisesti.

```
// Make the service available.
BLE.addService(environmentalSensingService);
BLE.setConnectable(true);
Serial.println("Advertising services.");
BLE.advertise();
// Turn off LED to indicate startup is complete.
digitalWrite(LED_BUILTIN, LOW);
```

Kuvio 15. BLE-palvelun mainostaminen

4.1.3 Datan lähetys

Arduino Nanoon tulee ottaa yhteys Raspberry Pi:ltä. Varmistuksena BLE-yhteydestä Arduinossa palaa vihreä valo merkkinä siitä, että yhteys on muodostettu ja arvoja voidaan alkaa kirjoittamaan antureilta mainostettavaan palveluun. Anturilukemat talletetaan muuttujiin, jotka kirjoitetaan niille tarkoitettuihin tuntomerkkeihin palvelussa. Palveluun kirjoittaminen tapahtuu funktiolla *{Chararacteristic_name}.writevalue((uint16_t) round(temperature * 100))*. Kuviossa 16 esitellään arvon kirjoittaminen palveluun.

```
Serial.print("Pressure: ");
Serial.print(pressure);
Serial.println("kPa");
pressureCharacteristic.writeValue((uint32_t) round(pressure * 10000)); //

Serial.print("Temperature: ");
Serial.print(temperature);
Serial.println("°C");
temperatureCharacteristic.writeValue((int16_t) round(temperature * 100));

Serial.print("Humidity: ");
Serial.print(humidity);
Serial.println("%");
humidityCharacteristic.writeValue((uint16_t) round(humidity * 100)); // St
```

Kuvio 16. Anturilukemien välittäminen BLE-palveluun

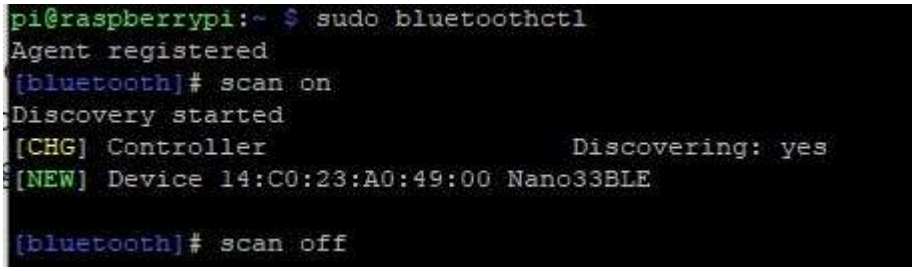
Anturilukemat välitetään heksadesimaali muodossa BLE:n yli Raspberry Pi:lle. Seuraavassa kappaleessa käsitellään, kuinka bittiluvut käännetään ihmiselle luettavaan muotoon eli desimaaleihin. Yhteyden katketessa Arduinolta led-valo sammutetaan ja BLE-palvelua aletaan taas mainostamaan.

4.2 Python-ohjelmointi

Tässä kappaleessa käydään läpi Python-ohjelmointi osuus. Pythonilla toteutettiin BLE-datan vastaanotto, datan käsittely sekä rajapinnan toteutus.

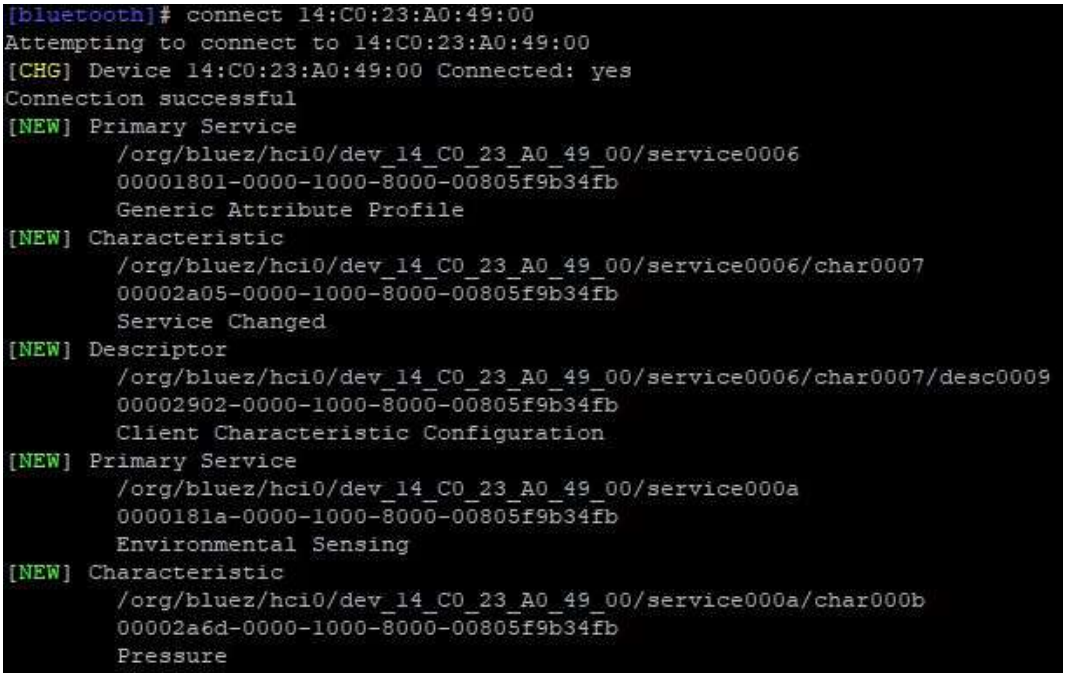
4.2.1 BLE-yhteyden testaaminen

Bluetooth Low-Energy yhteyttä Arduinolle voidaan testata Linuxin sisäisellä työkalulla bluetoothctl. Työkalulla muodostetaan testi yhteys Arduino Nanolle. Työkalun avulla voidaan myös tarkastaa laitteiden Bluetooth-laiteosoitteet.



Kuvio 17. Bluetoothctl-ohjelmalla suoritettu Bluetooth-laitteiden skannaus

Kuviossa 17 käynnistetään Raspberry Pi:llä bluetoothctl- ohjelma ja suoritetaan bluetooth-laitteiden skannaus. Kuviosta nähdään, että Arduino Nano 33 BLE on saatavilla ja yhdistettävissä.



Kuvio 18. BLE-testiyhteys Arduinolle

Kuviossa 18 on esitetty, miten voidaan muodostaa yhteys Arduino Nano 33 BLE-laitteeseen käyttäen bluetoothctl komentoja. Tällöin nähdään saatavilla olevat palvelut sekä palvelun tarjoamat ominaisuudet.

4.2.2 BLE-yhteys Arduino Nanoon

BLE-kommunikaatiossa Arduinon kanssa käytetään Pythonin moduulia bluepy. Moduuli toimii ainoastaan Linux-ympäristöissä. Raspberry Pi:n käyttöjärjestelmänä toimii Linux-pohjainen käyttöjärjestelmä Raspbian, joten bluepy moduuli on yhteensopiva BLE-kommunikaatioon Raspberry Pi:n kanssa. Kuviossa 19 on esitetty moduulin asennus Python 3:lle.

```
$ sudo apt-get install python3-pip libglib2.0-dev
$ sudo pip3 install bluepy
```

Kuvio 19. Bluepy-kirjaston asennus

Arduino Nanoon otetaan BLE-yhteys kirjastoa käyttäen asettamalla *Peripheral()* -luokalle parametrina BLE -laitteen mainostettu MAC-osoite kuvion 20 mukaisesti. Sen jälkeen annetaan *device.getServiceByUUID()* -luokalle parametrina halutun palvelun id, jonka arvoja halutaan vastaanottaa.

```
device = btle.Peripheral(input("Enter mac: "))
_ = device.services
sensors = btle.UUID("181A")
service = device.getServiceByUUID(sensors)
```

Kuvio 20. Bluepy-kirjaston käyttö

4.2.3 Datan käsittely

BLE-yhteyden kautta vastaanotetut anturilukemat eivät ole vielä vastaanotettaessa luettavassa muodossa, sillä ne ovat bitteinä. Ohjelmassa tieto käännetään luettavaan muotoon kahdella vaihtoehtoisella tavalla. Tässä kappaleessa käydään läpi kaksi esimerkkiä, miten data käännetään Python-ohjelmakoodissa desimaalimuotoon.

```
# Reading temperature and conversion to int from bytes. Byte order least significant
def read_temperature(service):
    localtime = time.asctime(time.localtime(time.time()))
    temperature_char = service.getCharacteristics("2A6E")[0]
    temp_sensor_value = temperature_char.read()
    temperature = (int.from_bytes(temp_sensor_value,
                                  byteorder="little", signed=True)/100)

    temp_dic = {"name": "temperature", "value": temperature, "time": localtime}
    response = requests.post(
        "http://127.0.0.1:5000/delektre/sensors", json=temp_dic)
    return temperature
```

Kuvio 21. Lämpötilan kääntö desimaaliluvuksi

Kuviossa 21 on funktio *read_temperature()*, joka saa parametrina BLE-laitteen palvelun id:n. Ensimmäiseksi funktiossa haetaan BLE-palvelusta haluttu tieto (Characteristic), joka pitää sisällään lämpötila-anturin lukeman. Lämpötila haetaan bluepy kirjastoa käyttäen *service.getCharacteristics()[0]* funktiota käyttäen. Kyseiselle funktiolle annetaan merkkijono 2A6E, joka vastaa BLE-palvelussa olevaa lämpötilatiedon id:tä. Tämä bittijono talletetaan muuttujaan *temp_sensor_value* heksadesimaali muodossa. Pythonilla heksadesimaaliarvo käännetään desimaaliluvuksi komennolla *(int.from_bytes(temp_sesor_vlue,bteoder="little",signed=True)/100)*. Komento muuttaa bitit desimaaliksi ottaen huomioon bittien järjestyksen ja jakamalla luvulla 100.

Vaihtoehtoinen tapa sensorilukeman kääntämiseen ihmiselle luettavaan muotoon on käyttää Python-kirjastoa struct. Kirjastoa voidaan käyttää binääridatan käsittelemiseen tiedostoissa tai kuten tässä tapauksessa BLE-yhteyden ylitse tulevien bittien muuntamiseen Python arvoiksi. Muuntaminen tapahtuu kuvion 22 mukaisesti.

```
# Using struct library for conversion
def read_accelometer(service):
    localtime = time.asctime(time.localtime(time.time()))
    acce_x_char = service.getCharacteristics("0001")[0]
    acce_y_char = service.getCharacteristics("0002")[0]
    acce_z_char = service.getCharacteristics("0003")[0]

    acce_x_sensor_value = acce_x_char.read()
    acce_y_sensor_value = acce_y_char.read()
    acce_z_sensor_value = acce_z_char.read()

    accelometer_data_x = (
        struct.unpack('f', acce_x_sensor_value))
    accelometer_data_x = accelometer_data_x[0]/100

    accelometer_data_y = (
        struct.unpack('f', acce_y_sensor_value))
    accelometer_data_y = accelometer_data_y[0]/100

    accelometer_data_z = (
        struct.unpack('f', acce_z_sensor_value))
    accelometer_data_z = accelometer_data_z[0]/100
```

Kuvio 22. Kääntäminen käyttäen *struct* -kirjastoa

Kuviossa 22 suoritetaan kiihtyvyysanturi lukemalle kääntö käyttäen *struct*- kirjastoa. Funktio on muuten toiminnallisuuksiltaan samanlainen kuin kuviossa 21, mutta kääntö suoritetaan käyttäen komentoa *struct.unpack()*.

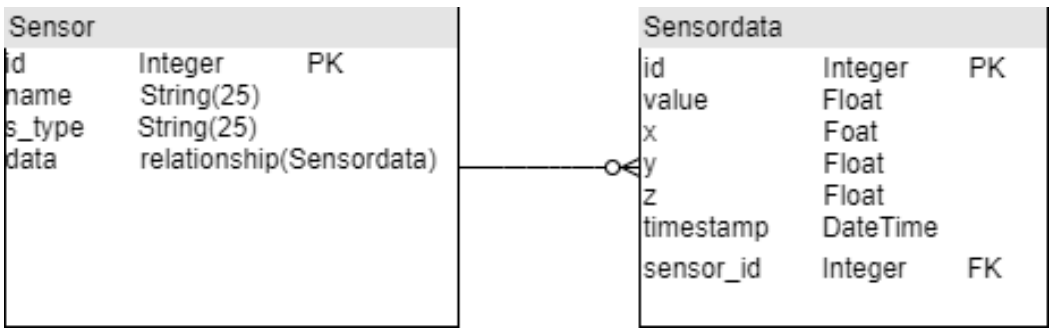
4.2.4 Rajapinnan toteutus

Rajapinnan tärkeimpänä ominaisuutena tulee olla tiedon saaminen reaaliajassa antureilta. GET-menetelmää käyttäen voidaan hakea sensoreiden dataa. Anturilukemien välitykseen rajapinnalle käytetään POST-menetelmää, jolla lähetetään viimeisimmät anturilukemat rajapinnalle. Rajapinnan toteutukseen valikoitui Flask sen skaalautuvuuden sekä monipuolisten lisäkirjastojen vuoksi. Flaskissa on myös tuki Pythonin omille lisäosille.

Rajapinnan alkukehitys vaiheissa lähdettiin toteuttamaan toiminnallista rajapintaa, mistä olisi saatavilla vain viimeisimmät sensorilukemat. Myöhemmässä vaiheessa rajapintaan lisättiin tietokanta, mikä mahdollistaa anturilukemien tallettamisen. Tällä laajennuksella rajapinnalta voidaan myös hakea vanhempaa dataa antureilta. Rajapinta toteutettiin tutkimuksen kannalta kahdella eri tietokannalla. Tietokantana toimii SQLite. Tässä kappaleessa perehdytään rajapintatoteutukseen, jossa tietokantana toimii SQLite, sen ollessa optimaalisempi vaihtoehto rajapinnalla.

Rajapintaa toteutettaessa rajapinnalle sisällytetään flask, sekä muut mahdolliset kirjastot, joita toteutus vaatii. Tietokantana rajapinnalla toimii SQLite, jonka käyttöön ja käsittelyyn sopii SQLAlchemy, joka on Pythonin SQL-lisäosa.

Rajapinnalle täytyy toteuttaa tietokantamallina toimivat luokat. Tietokantaan luodaan kaksi taulua, ensimmäinen sensoreita ja niiden nimiä varten ja toinen sensorien dataa ja niiden aikaleimoja varten. Tietokannan rakenne on esitetty kuviossa 23.



Kuvio 23. SQLite tietokantakaavio

Sensor taulussa sensorien tiedot ja Sensordata-taulussa antureiden lukemat ja aikaleimat näille. ID toimii pääavaimena (primary key) ja Sensordata-taulussa sensor_id viiteavaimena (foreign key).

```
class Sensor(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String(25))
    s_type = db.Column(db.String(25))
    data = db.relationship('Sensordata', backref='sensordata', lazy='dynamic')

class Sensordata(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    value = db.Column(db.Float)
    x = db.Column(db.Float)
    y = db.Column(db.Float)
    z = db.Column(db.Float)
    timestamp = db.Column(db.DateTime, default=datetime.datetime.now)
    sensor_id = db.Column(db.Integer, db.ForeignKey('sensor.id'))
```

Kuvio 24. SQLite tietokannan malliluokat

Kuviossa 24 esitetty tietokannan malliluokat, joiden avulla tietokantaa käsitellään. Anturin id:n perusteella voidaan hakea anturin dataa.

Rajapinta toteutettiin käyttäen Flask-RESTful -kirjastoa. Kirjaston avulla luodaan luokka ja sinne useita HTTP-metodeja. Kirjaston avulla on helppoa ja yksinkertaista tehdä GET, POST, PUT ja DELETE metodit kaikki käyttäen samaa URL-osoitetta. Rajapinnalle toteutettiin kaksi luokkaa, joista kummallakin on omat toimintonsa tietokannan käsittelyyn. Rajapinnalla toteutettiin *SensorCollection* -luokka, jonka tehtävä on hakea anturien tiedot sekä lisätä uusia antureita. Luokan toiminnallisuudet ovat taulukon 3 mukaiset.

Taulukko 3. SensorCollection -luokan toiminnallisuudet.

URL	**HTTP-metodi**	**Toimenpide**
delektre/sensors	GET	Palauttaa kaikkien anturien tiedot.
delektre/sensors	POST	Lisätään uusi anturi tietokantaan.

Taulukon 3 mukaisesti luokkaan toteutettiin GET-metodi, jolla haetaan kaikkien sensorien tiedot sekä POST-metodi, jolla lisätään uusia antureita tietokantaan.

```
class SensorCollection(Resource):
    def get(self):
        sensors = Sensor.query.all()
        return jsonify(sensors_schema.dump(sensors))

    def post(self):
        sensor = request.get_json()
        sensor_name = sensor["name"]
        sensor_type = sensor["type"]
        sensor_db = Sensor(name=sensor_name, s_type=sensor_type)
        db.session.add(sensor_db)
        db.session.commit()
        return jsonify(sensor)
```

Kuvio 25. SensorCollection -luokka

Kuviossa 25 on kuvattu *SensorCollection* -luokka, jonka kautta voidaan hakea kaikki anturit tietokannalta GET-metodilla sekä POST-metodi, jonka avulla tietokantaan lisätään uusia antureita.

```
api.add_resource(SensorCollection, '/delektre/sensors')
```

Kuvio 26. Lisätään *SensorCollection* -luokalle URL-osoite.

Kuviossa 26 linkitetään kuviossa 23 esiintyvä *SensorCollection* -luokka URL-osoitteeseen. Tähän URL-osoitteeseen voidaan tehdä GET -ja POST pyyntöjä.

Rajapinnalla luotiin toinen luokka, jonka kautta voidaan hakea tietokannalta anturin lukemat, lisätä anturi lukemia sekä poistaa antureita tietokannalta. Luokan kautta on mahdollista suorittaa taulukon 4 mukaisia toimenpiteitä.

Taulukko 4. SensorItem -luokan toiminnot

URL	HTTP-metodi	Toimenpide
delektre/sensors/{id}	GET	Palauttaa id:n mukaisen sensorin datan.
delektre/sensors/{id}	POST	Uuden sensorilukeman lisäys id:n mukaiselle sensorille.
delektre/sensors/{id}	DELETE	Poistetaan id:n mukainen sensori tietokannalta.

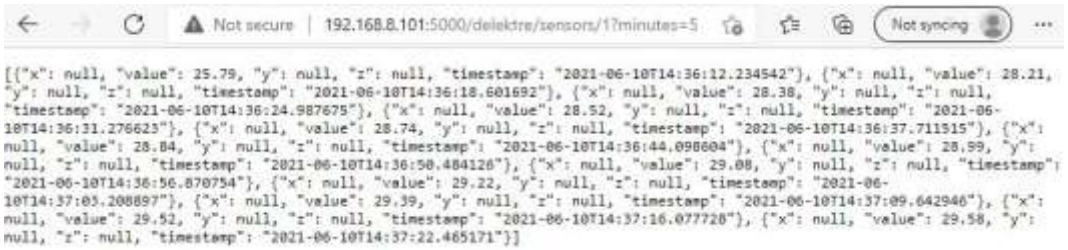
Taulukon 4 mukaisesti *SensorItem* -luokan avulla on mahdollista hakea anturin dataa id:n perusteella käyttäen GET-metodia. POST-metodilla lähetetään anturilukemia tietokannalle. Delete-metodilla voidaan poistaa antureita tietokannasta, mikäli nämä eivät ole enää aktiivisia tai käytössä.

Lisäksi *SensorItem* -luokalle toteutettiin ominaisuus hakea anturidataa tietokannasta minuuttien, tuntien tai päivien perusteella. Näin tiedon määrää voidaan skaalata ja hakea tietokannalta se tieto, mikä on tärkeää.

```
class SensorItem(Resource):
    def get(self, id):
        sensor = Sensor.query.get(id)
        if sensor:
            if "minutes" in request.args:
                minutes_ = int(request.args.get('minutes'))
                filtertime = datetime.datetime.now() - timedelta(minutes = minutes_)
                asd = Sensordata.query.filter
                (Sensordata.timestamp >= filtertime, Sensordata.sensor_id == id).all()
                return datas_schema.dump(asd)
```

Kuvio 27. SensorItem -luokka

Kuviossa 27 on esimerkki *SensorItem* -luokka tehdystä toteutuksesta, miten tietokannalta haetaan anturidataa minuuttien perusteella. Haettaessa dataa ajan perusteella sisällytetään URL-osoitteeseen käytettävät argumentit. URL-osoite voisi tässä tapauksessa olla http://127.0.0.1:5000/delektre/sensors/1?minutes=5 .



Kuvio 28. 5minuutin mittausten hakutulokset lämpötila-anturilta

Kuviossa 28 on tehty GET-pyyntö rajapinnalle 5 minuutin hakutulosten saamiseksi. Rajapinta palauttaa vastauksena anturilukemat sekä aikaleimat JSON-muodossa.

5 YHTEENVETO

Työn tarkoituksena oli kerätä Arduino Nano 33 BLE Sense:ltä anturidataa ja lähettää ne Bluetooth Low Energyä käyttäen Raspberry Pi:lle tiedon käsittelyyn Pythonilla tehdylle sovellukselle. Tehtävänä oli luoda rajapinta, jonka kautta anturilukemat voidaan hakea asiakasohjelmalla, tässä tapauksessa WordPress:llä tarkasteltavaksi ja analysoitavaksi. Opinnäytetyön tuloksena voidaan toteuttaa IoT-järjestelmä, jossa hyödynnetään BLE-teknologiaa sekä rajapintaa, joka mahdollistaa anturidatan esittämisen monissa erilaisissa asiakasohjelmissa.

Projektissa keskityttiin lähinnä BLE-teknologiaan tiedonvälityksessä sekä rajapinnan toteutukseen käyttäen Python-ohjelmointikieltä, mutta projekti antoi myös hyvän kuvan siitä, millainen IoT-järjestelmä on, mitä erilaisia teknologioita siinä voidaan käyttää, mitä tulee ottaa huomioon tietokantaa suunniteltaessa sekä hyvän yleiskäsityksen IoT-järjestelmien arkkitehtuurista.

Projektin lopputulos on toimiva ja mahdollistaa anturidatan talletuksen ja jatkokäytön muissa ohjelmissa käyttäen kehitettyä rajapintaa. Kehitysmahdollisuuksia projektissa voisi olla tarjota rajapinnan kautta tuleva data selkeämmässä struktuurissa sekä lisätä sinne muun muassa anturin omistajayritys ja sijainti, mikä mahdollistaisi järjestelmän skaalautumisen suurempiin järjestelmiin ja antureiden seuranta helpottuisi. Mikäli projektia lähdettäisiin skaalaamaan suurempaan mittakaavaan tietokannaksi soveltuvampi vaihtoehto voisi olla jokin NoSQL-tietokanta, kuten MongoDB. Tähän toteutukseen SQLite on kuitenkin hyvä vaihtoehto.

LÄHTEET

/1/ Altexsoft, IoT- järjestelmän arkkitehtuuri. Viitattu 11.06.2021. https://www.altexsoft.com/blog/iot-architecture-layers-components/

/2/ Delektre Oy. Viitattu 10.05.2021. https://www.linkedin.com/company/delektre-oy

/3/ Arduino Nano 33 BLE Sense. Viitattu 11.04.2021. https://store.arduino.cc/arduino-nano-33-ble-sense-with-headers

/4/ Raspberry Pi. Viitattu 11.04.2021. https://www.linux.fi/wiki/Raspberry_Pi

/5/ Arduino IDE. Viitattu 01.04.2021. https://www.arduino.cc/en/Guide/Environment

/6/ Bluetooth Low Energy. Viitattu 01.04.2021. https://www.bluetooth.com/bluetooth-resources/intro-to-bluetooth-low-energy/

/7/ Generic Access Profile. Viitattu 13.04.2021. https://learn.adafruit.com/introduction-to-bluetooth-low-energy/gap

/8/ Generic Attribute Profile. Viitattu 11.04.2021. https://www.oreilly.com/library/view/getting-started-with/9781491900550/ch04.html

/9/ Python-ohjelmointikieli. Viitattu 12.05.2021. https://fi.wikipedia.org/wiki/Python_(ohjelmointikieli)

/10/ Wikipedia ohjelmointirajapinta. Viitattu 01.06.2021. https://fi.wikipedia.org/wiki/Ohjelmointirajapinta

/11/ REST, Wikipedia. Viitattu 01.05.2021. https://fi.wikipedia.org/wiki/REST

/12/ JSON. Viitattu 05.07.2021. https://datatracker.ietf.org/doc/html/rfc8259

/13/ Flask ohjelmistokehitys. Viitattu 11.05.2021. https://www.fullstackpython.com/flask.html?web=1&wdLOR=c4A251EA5-C1A9-4529-84AE-300345CDF457

/14/ Flask-RESTful kirjasto. Viitattu 20.03.2021. https://flask-restful.readthedocs.io/en/latest/

/15/ RRDTool. Viitattu 10.05.2021. https://oss.oetiker.ch/rrdtool/tut/rrdtutorial.en.html

/16/ SQLite. Viitattu 10.01.2022. https://fi.wikipedia.org/wiki/SQLite

/17/ ArduinoBLE. Viitattu 11.04.201. https://www.arduino.cc/en/Reference/ArduinoBLE

/18/ ArduinoHTS221 kirjasto. Viitattu 10.04.2021. https://www.arduino.cc/en/Reference/ArduinoHTS221

/19/ ArduinoLPS22HB kirjasto. Viitattu 11.05.2021. https://www.arduino.cc/en/Reference/ArduinoLPS22HB

/20/ ArduinoAPDS9960 kirjasto. Viitattu 22.04.2021. https://www.arduino.cc/en/Reference/ArduinoAPDS9960

/21/ ArduinoLSM9DS1 kirjasto. Viitattu 03.05.2021. https://www.arduino.cc/en/Reference/ArduinoLSM9DS1

/22/ PDM Arduino kirjasto. Viitattu 08.04.2021. https://www.arduino.cc/en/Reference/PDM

/23/ bluepy, IanHarvey. Viitattu 22.05.2021. http://ianharvey.github.io/bluepy-doc/

/24/ Collin J & Saarelainen A, Teollinen internet, III OSA Datasta älyä, Tietovarasto. Viitattu 29.05.2021.