VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES

Antero Joensuu

IOT Datalogger

Case Avalus Oy

Liiketalous
2017

VAASAN AMMATTIKORKEAKOULU
Tietojenkäsittely

TIIVISTELMÄ

Tekijä	Antero Joensuu
Opinnäytetyön nimi	IOT Datalogger Case Avalus Oy
Vuosi	2017
Kieli	suomi
Sivumäärä	30
Ohjaaja	Klaus Salonen

Monissa vapaa-ajan asunnoissa vietetään hyvin vähän aikaa talven aikana, jolloin riski eri vahingoille on suurimmillaan. Jos vapaa-ajan asunnon lämpötila pääsee laskemaan liian alas lämmitysjärjestelmään tulevan vian vuoksi, se voi tuoda mukanaan vesiputkien jäätymisen ja sitä kautta niiden repeämisen. Vahingot tämän sattuessa voivat olla mittavat.

Riskien minimoimiseksi toteutin lämpötilatietoa keräävän laitteen, jota voi seurata Internetin välityksellä. IoT-laite lähettää lämpötilatietoa MQTT välityspalvelimen kautta tietokantaan, josta www-sivu poimii tiedot nähtäville. Opinnäytetyössä käydään läpi projektin toteutus, sekä projektin teoriapohjaa.

Työn tuloksena syntyi tavoitteiden mukainen laite, joka soveltuu erinomaisesti sille suunniteltuun tehtävään. Vaikka opinnäytetyön lämpötilavahti on kohdistettu vapaa-ajan asuntoihin, sitä voi käyttää moniin muihinkin käyttötarkoituksiin.

Työni toimeksiantajana on Avalus Oy, joka on pieni ohjelmistoalan yritys. Se on perustettu vuonna 2013 ja se toteuttaa erilaisia tulostuksia sekä IoT-laitteita ja -ohjelmistoja.

Avainsanat Internet of Things, Ohjelmointi, Arduino

VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES
Tietojenkäsittelyn koulutusohjelma

ABSTRACT

Author	Antero Joensuu
Title	IOT Datalogger Case Avalus Oy
Year	2017
Language	Finnish
Pages	30
Name of Supervisor	Klaus Salonen

In wintertime summer cottages are often unoccupied for several months which increases the risk for different damages. If the inside temperature decreases due to heating system failures, the damages can be massive. For example, water damage is possible because water pipes may freeze when the temperature is low. These problems can cause expensive repairs in the cottage.

To minimize risks, I built a device which collects temperature information and shows them on a web site. An IoT-device sends the temperature data via MQTT-broker to a database and shows the data clearly on the web page.

The Results of the project were promising, and the device fits perfectly for the intended purpose. Although the temperature follower is planned for cottages, it could be used for many different purposes.

The client of this thesis is Avalus Oy, which is a small programming company. It was founded in 2013, and it implements printouts as well as Internet of Things devices and -software.

Keywords Internet of Things, Programming, Arduino

SISÄLLYS

KUVA- JA TAULUKKOLUETTELO

LYHENTEET JA TERMIT

Arduino	Elektroniikka-alusta
IDE	(Integrated Development Environment) Integroitu ohjelmointiympäristö
C/C++	Ohjelmointikieli
IoT	The Internet of Things; Esineiden Internet
MQTT	Viestintäprotokolla
PHP	PHP: Hypertext Preprocessor, ohjelmointikieli
Python	Monipuolinen ohjelmointikieli
SQL	Structured Query Language, kyselykieli

1 JOHDANTO

Monissa vapaa-ajan asunnoissa ollaan hyvin vähän talven aikana, jolloin riski suurille vahingoille on todellinen. Jos vapaa-ajan asunnon sisälämpötila pääsee jonkin vian vuoksi laskemaan liian alas, se voi tuoda mukanaan monia ongelmia. Esimerkiksi vesiputkien jäätyminen ja sen myötä putken halkeaminen aiheuttaa pahimmillaan mittavat vahingot. Tästä syntyi idea toteuttaa ongelmaan ratkaisu: laite, joka jakaa lämpötilatietoa Internet-sivuille, josta se on luettavissa etänä.

Tässä opinnäytetyössä käsitellään projektiin liittyvien asioiden ympäriltä teoriaa, joka koostuu johdannosta Esineiden Internetiin (IoT), käytetyistä ohjelmointikielistä sekä projektin kokonaisrakenteesta. Lopuksi työssä käydään vaihe vaiheelta projektin toteutus läpi.

Kehitysalustana projektissa toimii LoLin NodeMcu V3, johon kytketään DHT22-anturi. Projektissa käytetään monia ohjelmointikieliä. Esimerkiksi alustan ohjelmoinnissa käytetään ohjelmointikieltä, joka perustuu alun perin C/C++-kieleen.

Projektiin lähtiessä kokemukseni siihen liittyvistä ohjelmointikielistä oli melko vähäistä. Se ei haitannut, sillä Arduinoon liittyvissä projekteissa on todella hyvin saatavilla ohjeita sen ympärillä olevan suuren yhteisön vuoksi. Ennen projektia tutuimpia ohjelmointikieliä minulle olivat C#, SQL sekä HTML.

1.1 Toimeksiantaja

Toimeksiantajana opinnäytetyölleni toimii pieni alajärveläinen ohjelmistoalan yritys Avalus Oy. Yritys on perustettu vuona 2013 ja se työllistää kaksi henkilöä. Ohjelmointipalveluiden lisäksi yritys tarjoaa tulostuspalveluita.

1.2 Tavoitteet

Työn tavoitteena on toteuttaa IoT-projekti alusta loppuun, jolloin saa kokonaisvaltaisen kuvan Esineiden Internetin maailmasta. IoT-laitteen tulee mitata ja lähettää lämpötilatietoa. Tämän lisäksi tarvitaan Internet-sivu, jossa on selkeästi nähtävillä seuraavat tiedot: laitekohtainen id, lämpötilatieto sekä aikaleima. Aikatavoitteena projektin toteutuksella on 2017 vuodenvaihde.

2 INTERNET OF THINGS

Kevin Ashton käytti ”Internet of Things”- fraasia ensimmäisen kerran jo vuonna 1999. Hän tarkoitti sillä tietokoneiden ja laitteiden yhdistämistä Internettiin, sekä niiden ohjausta ja kommunikointia sen välityksellä. Todellisuudessa IoT on ollut tiedossa jo sitä aiemmin, mutta sitä ennen sillä ei ollut vielä yhteisesti käytettyä nimeä. (Norris, 2015) Kevin Ashtonia voidaan siis pitää IoT ilmiön perustajana. Hänen mukaan ihmisen ei pitäisi luoda kaikkea dataa kuten aiemmin, sillä ihmisillä on rajallisesti aikaa ja tarkkuutta valtavan datamäärän keskellä. Pyrkimyksenä siis on, että luodaan arkkitehtuuri, jossa laitteet voisivat kerätä ja analysoida tietoa ilman ihmisten apua. (Ashton, 2009)

Asioiden ja esineiden Internet on yksi suurimpia tulevaisuuden mahdollisuuksia. Pienlaitteiden ja tietokoneiden kytkeminen verkkoon tulee helpottamaan ja nopeuttamaan koko ajan enemmän monia prosesseja. IoT:tä onkin pidetty seuraavana suurena informaatiovallankumouksena Internetin tulon jälkeen. (Chan, 2015)

Puhelimet, tietokoneet ja tabletit ovat tällä hetkellä meille tutuimpia- ja yleisimpiä laitteita, jotka ovat yhteydessä Internettiin. IoT sisältää kuitenkin paljon muuta, kuin nämä yleisimmät laitteet. Myös semmoiset tavarat ja tuotteet, joita aiemmin ei olla pidetty juurikaan teknisinä, ovat yhteydessä Internetiin. Sensoreiden sekä mikroprosessoreiden kautta mm. vaatteet, kodinelektroniikka ja ajoneuvot ovat osa maailmanlaajuista verkkoa. Sensorit mittaavat lähes kaikkea niiden ympäristössä tapahtuvaa, kuten esimerkiksi lämpötilaa ja liikettä. (Chan, 2015)

3 KÄYTETYT OHJELMOINTIKIELET

Projektin toteuttamiseen vaaditaan usean ohjelmointikielen ymmärtämistä ja kirjoittamista. Alustassa käytettiin C++ pohjautuvaa ohjelmointikieltä, web-osuus toteutettiin PHP/HTML kielillä, MySQL-tietokanta SQL-kielellä ja MQTT brokerin ja tietokannan välinen rajapinta toteutettiin Pythonilla. Seuraavaksi käydään läpi perusasioita näistä työhön liittyvistä ohjelmointikielistä.

3.1 Arduinon ohjelmointikieli

Arduino-laitteiden ohjelmointikieli perustuu alun perin C++-ohjelmointikieleen. Se on C++:sta yksinkertaistettua ohjelmointikieltä, jonka Arduino IDE tulkkaa C-kielelle. Tämän jälkeen C-kieli käännetään mikrokontrollerin ymmärtämälle kielelle. Yksinkertaisuutensa vuoksi Arduinon ohjelmointikieltä pidetään helposti opittavana. (Banzi, 2011)

3.1.1 C++-kieli

C++ on yksi tunnetuimmista olio-kielistä. Kielen perustana on C-kieli, johon on tehty joitakin lisäyksiä, kuten olio-ohjelmointimekanismit sisältävä laajennus. Olio-ohjelmointikielen ominaisuuksien lisäksi muita C++-kielen ominaisuuksia ovat viittausmuuttujat (references), aliohjelmien kuormitus (overloading), aliohjelmaparametrien oletusarvot ja poikkeuskäsittely (exception handling). C++-ohjelmointikieli on myös saanut monia vaikutteita muista ohjelmointikielistä. Esimerkiksi luokka-käsite, periytyminen ja virtuaaliset aliohjelmat ovat peräisin Simula67:sta (Hietanen, 2008)

C++-kielessä on ollut monia kehitysvaiheita. Alkuperäinen perustaja kielelle on Bjarne Stroustrup, mutta monet muut henkilöt ovat olleet myös kehittämässä sitä.

C++-kielen tärkeimpiä vaiheita:

- 1980, C with Classes. Bjarne Stroustrup kehitti kielen, jolla voidaan tuottaa nopeita ja vähän tilaa vieviä tapahtumaohjattuja simulointiohjelmia.
- 1983, C++. Rick Mascitti keksi kielelle nimen.
- 1989, C++ 2.0. Kieleen lisättiin moniperiytyminen, parametrisoidut tyypit ja poikkeuskäsittely
- 1990, The Annotated C++ Reference Manual. C++ 2.1. Teos sisältää epävirallisen standardin, joka hyväksyttiin virallisen standardiehdotuksen pohjaksi.
- 1994/1995, Lisäyksiä standardiehdotukseen mm. ajonaikaiset tyyppitarkistukset.
- 1997, C++ standardi (ISO/ANSI C++ Draft) Standardi hyväksyttiin 14. marraskuuta 1997. Se sisältää C++ kielen ja sen standardikirjaston.
  (Hietanen, 2008)

Esimerkki, joka tulostaa ”Hello world!” C++-kielellä:

```
#include <iostream.h>
        int main(void)
        {
          cout << "Hello world!" << endl;
          return 0;
        }
```

3.2 Python-kieli

Python on selkeäsyntaksinen ja yleisesti helposti opittava ohjelmointikieli, jonka takia se sopii hyvin ensimmäiseksi ohjelmointikieleksi. Usein Pythonia käytetään tietoliikenne- ja ylläpitosovellusten kirjoittamisessa, mutta siihen on saatavilla laajennuksia, joiden avulla sitä voi käyttää myös muihin tarkoituksiin. (Rasila, 2004) Se on tulkattava, interaktiivinen olio-ohjelmointikieli, jossa on rajapinnat moniin eri järjestelmiin ja tietokantoihin. Esimerkiksi tässä projektissa Pythonia käytetään MQTT brokerin sekä tietokannan välisessä kommunikoinnissa. Se on laajennettavissa C- ja C++-ohjelmointikielillä, ja sitä voi myös sisällyttää C- ja C++ kielisiin ohjelmiin skriptikielenä. Python on myös käytettävissä monissa eri alustoissa, sillä se toimii useimmissa Unix varianteissa, Mac- sekä Windows tietokoneilla. (Python, 2017)

Esimerkki, joka tulostaa ”Hello world!” Python-kielellä:

```
print ’Hello world!’
```

3.3 PHP-kieli

PHP: Hypertext Preprosessor eli PHP on kieli, jota käytetään ensisijaisesti Web-sivujen luomisessa. Se on komentosarjakieli, jossa koodi tulkitaan vasta suoritusvaiheessa. Koska PHP-koodi suoritetaan vasta palvelimella ennen kuin nettisivut lähetetään selaimelle, sillä voi käsitellä palvelimen tiedostoja ja tietokantoja. PHP-kieli on käytettävissä lähes kaikissa webhotelleissa. Muita vahvuuksia siinä on helppokäyttöisyys sekä monipuolisuus. Kieltä kirjoittaessa tulee toimia tarkasti ja yksityiskohtaisesti, sillä PHP:n pitää olla täysin oikein kirjoitettua, toisin kuin esimerkiksi HTML, jossa pienet virheet tulkitaan jollain järkevällä tavalla. (Laaksonen, 2011)

Esimerkki, joka tulostaa ”Hello world!” PHP-kielellä:

```
<?php
echo ”Hello world!”;
?>
```

3.4 SQL-kieli

SQL muodostuu sanoista Structured Query Language eli strukturoitu kyselykieli. Vaikka nimi niin viittaa, SQL ei ole ainoastaan kyselykieli, vaan se muun muassa kattaa myös tietokannan rakenteen määrittelyn ja muuttamisen, päivitykset eli lisäykset, muutokset ja poistot, tapahtumankäsittelyn ohjaamisen sekä valtuuksien ja turvallisuuden hoidon. (Hovi, 2004)

SQL-käskyt ovat myös upotettavissa ohjelmointikieleen sekä sovellus- tai raporttikehittimeen. Tätä kutsutaan upotetuksi SQL:ksi. Tällaisessa tapauksessa vastaukset siirretään tietokannasta ohjelmointikielen muuttujille. Käskyjä voi muodostaa myös dynaamisesti ohjelmassa, ja lähettää sitten generoitu SQL käskytietokantajärjestelmälle käännettäväksi ja suoritettavaksi. (Hovi, 2004)

SQL:llä ei kerro, miten tietokantaoperaatiot tehdään. Sillä kerrotaan vain mitä haluttaan tehtävän. Se on siis ei-proseduraalinen kieli, jonka kysely tuo tyypillisesti tulosjoukossa joukon rivejä. Proseduraalinen ohjelma käsittelee tietoja tietue kerrallaan. (Hovi, 2004)

Yksinkertainen kysely asiakastietokannasta käyttäen SQL-kieltä:

```
select etunimi, sukunimi
from asiakas
```

4 IOT DATALOGGERIN RAKENNE

Datalogger eli tiedonkerääjä koostuu NodeMcu alustasta, joka sisältää ESP8266 WiFi-moduulin. Alustaan on kytketty myös DHT22-anturi, joka mittaa lämpötilaa. NodeMcu lähettää lämpötilatiedon MQTT-välityspalvelimen kautta MySQL-tietokantaan, josta web-sivu hakee tiedot selkeästi näkyville.

4.1 Arduino

Arduino on suosittu fyysisen tietojenkäsittelyn alusta, joka perustuu yksinkertaiseen piirilevyyn, jolla on sisään- ja ulostuloliitännät. Se koostuu kahdesta pääosasta. Ensimmäinen osa on Arduino-piirilevy, eli fyysinen alusta. Toinen osa on Arduino IDE eli ohjelmointiympäristö, jota hallitaan tietokoneella. (Banzi, 2011)

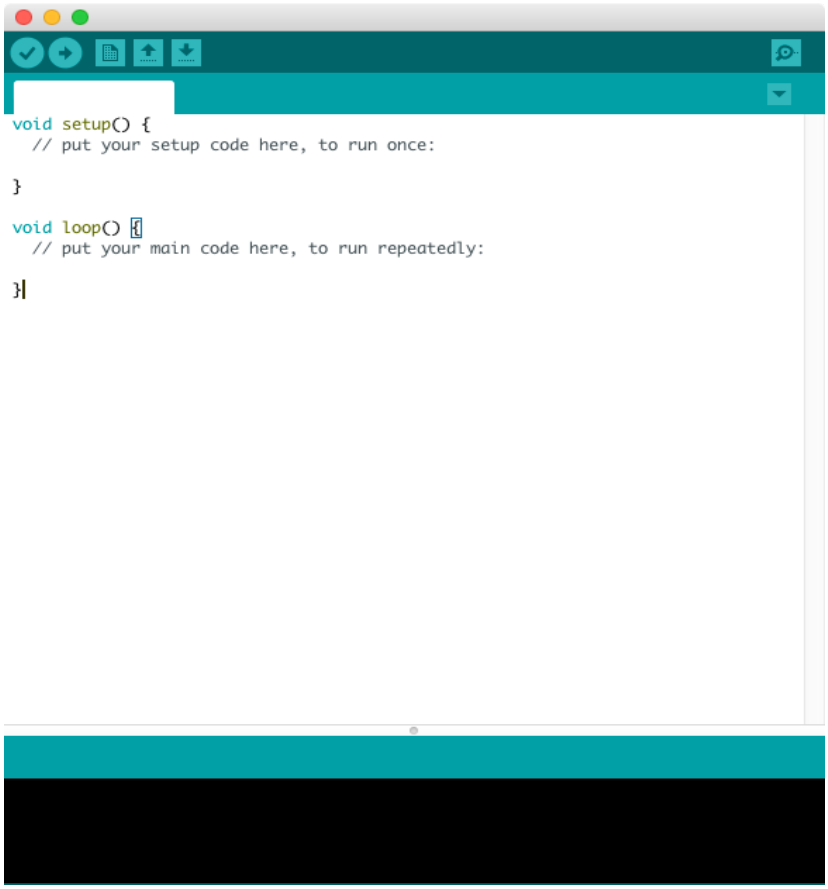
4.1.1 Arduino-piirilevy

Arduino piirilevy eli yksinkertainen mikrokontrollerikortti, joka sisältää pienen mikropiirin, muodostaa tietokoneen. Teholtaan tämä yksinkertainen kortti ei ole ”normaalin” tietokoneen tasolla, mutta se on käyttökelpoinen monenlaisten laitteiden rakenteluun. Arduino piirilevyn voit ohjelmoida tekemään haluamiasi asioita Arduino IDE:n avulla. (Banzi, 2011)

Arduino alustoja on todella paljon. Ne eroavat toisistaan muun muassa mikropiirin, piirilevyn koon- ja liitäntöjen määrän mukaan. Yleisimpiä Arduino alustoja ovat aloittelijoille hyvin sopivat Arduino Uno, sekä Arduino Mega, joka tarjoaa edistyneempiäkin ominaisuuksia. (Arduino, 2017)

4.1.2 Arduino IDE – Ohjelmistonkehitysympäristö

Arduino IDE on yksinkertainen kehitysympäristö Arduino laitteille. Aduino IDE:n pohjana on Processing IDE, ja se muistuttaakin ulkoisesti hyvin paljon sitä. Arduino IDE on ladattavissa https://www.arduino.cc/-sivustolta. Ladattu paketti tulee purkaa haluttuun kansioon, jolloin se on heti valmis käytettäväksi. Kortti liitetään USB-porttiin, jonka jälkeen se on valmis ohjelmoitavaksi. USB:n kautta kortille tulee ohjelman lataus sekä tulee käyttöjännite. (Arduino, 2017)



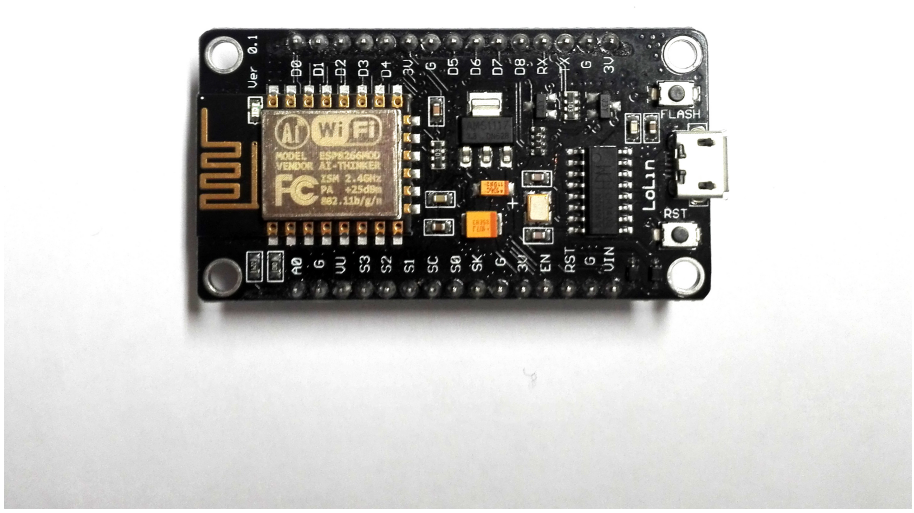
Kuva 1 - Arduino IDE Kehitysympäristö

Ohjelmoinnin aloittaminen:

1. Asetetaan USB:n käyttämä COM-portti (Työkalut – Portti – Sinun porttisi)
2. Aseta käytettävä korttityyppi (Työkalut – Kortti – Sinun korttisi)
3. Lataa esimerkkisovellus (Tiedosto – Esimerkit)

4.2 LoLin NodeMcu V3

NodeMcu on avoimen lähdekoodin IoT-alusta. IoT laitteessa tuli olla langaton verkkoyhteys, joten NodeMcu sopi siihen täydellisesti, sillä se rakentuu ESP8266 WiFi-moduulin päälle. Myös alustan pieni koko on iso etu tämän tyyppiseen laitteeseen. Sisään rakennetun Lua-tulkin ansiosta alustan ohjelmointiin on monia vaihtoehtoja. Tässä projektissa alustan ohjelmointi tapahtuu Arduino ympäristössä, jolloin käytetään NodeMcu kirjastoa.

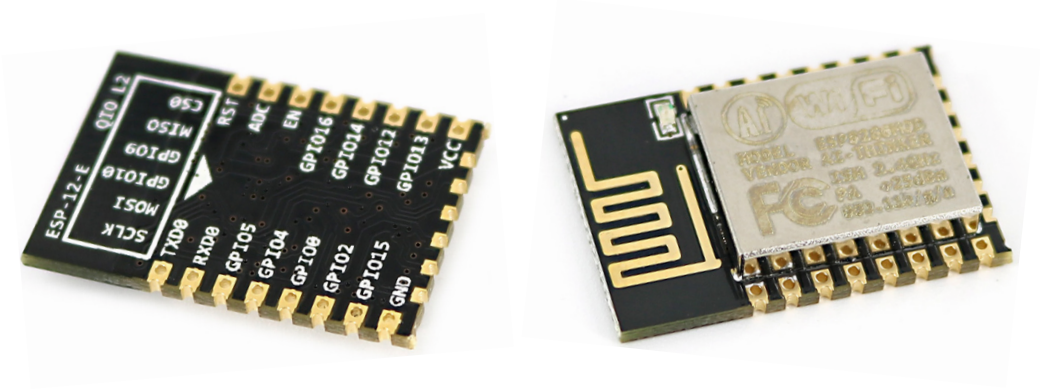


Kuva 2 - NodeMcu kehitysalusta

4.2.1 ESP8266 WiFi-moduuli

NodeMcu:n keskeisin osa on ESP8266 WiFi-moduuli. Se on vähän virtaa kuluttava ja itsenäisen mikrokontrollerin sisältävä WLAN-moduuli. ESP8266-moduulista on julkaistu useita versioita, ja projektissa käyttämässäni LoLin NodeMcu V3:ssa on käytetty ESP-12E versiota.

Moduulin kehitystyökalut ovat vapaasti saatavilla. Se onkin mahdollistanut ESP8266:n ympärille muodostuneen käyttäjä-/kehittäjäyhteisön. Moduuliin on kehitetty erilaisia sovelluksia ja sitä on käytetty moniin käyttötarkoituksiin. Sitä voidaan hyödyntää vaikkapa kevyenä web-palvelimena tai IoT-laitteena. (Benchoff, 2014)



Kuva 3 - ESP8266 ESP-12E

4.2.2 DHT22 anturi

Käytin projektissa DHT22-anturia, joka löytyy myös nimillä RHT03 ja AM2302. Se on pienikokoinen ja kevyt (2.4g) anturi, joka mittaa lämpötilaa sekä ilman kosteutta. Edullisen hinnan vuoksi se on myös yleisesti käytetty. DHT22 anturissa käyttöjännite eli Vcc on 3-5 V ja mittaustarkkuus kosteudelle asteikolla 0 - 100% on 2-5% ja mittaustarkkuus lämpötilalle 0,5 Celsiusastetta. Se sopii monenlaiseen käyttöön, myös ulos, sillä mittausskaala lämpötilalle on -40 celssiusasteesta +80 celssiusasteeseen. Mittaustaajuus on 0,5 Hz, eli mittaus voidaan suorittaa kerran kahdessa sekuntissa. (Adafruit, 2017)

DHT22- anturissa on neljä liitintä, joista se liitetään laitteeseen. (Taulukko 1) Ensimmäinen liitin on käyttöjännite (Vcc) ja toisen liittimen kautta menee tieto (DATA). Kolmas liitin on NC, eli se ei ole käytössä. Neljäs liitin on maadoitus (GND).

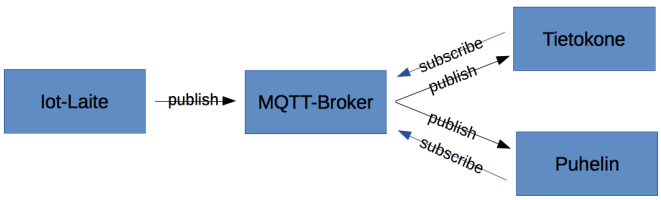
Selite	**Liitin**
Vcc	1
DATA	2
NC	3
GND	4

Taulukko 1 - DHT22 Liitäntä

4.3 MQTT

MQTT (MQ Telemetry Transport) on ”julkaise/tilaa”- viestintäprotokolla. Sen kehittäminen laitettiin alulle Andy Stanford-Clarkin ja Arlen Nipperin toimesta jo vuonna -99. MQTT-protokollaa käytetään nykyään monissa eri viestintäohjelmissa, esimerkiksi Facebook Messengerissä. Juuri puhelimen viestintään se sopii hyvin, koska keveytensä vuoksi Internetin- sekä akun käyttö on vähäistä. (MQTT, 2011) MQTT-protokolla aloitettiin rakentamaan silmällä pitäen seuraavia ominaisuuksia: yksinkertainen käyttöönottaminen ja toteutus, mukautuva tiedoneheyden varmistus, kevyt ja tehokas kommunikointi, sopivuus eri järjestelmiin sekä jatkuva tilan ylläpitäminen. Nämä tavoitteet ovat vielä nykypäivänäkin MQTT:n ydin. Protokolla on binäärinen, joten tieto siirretään konekielellä ja tämän myötä pakettikoko on todella pieni. (HiveMQ, 2015)

MQTT-protokolla perustuu siis julkaise/tilaa menetelmään. Laite, joka lähettää viestiä on julkaisija (Publisher). Laite joka vastaanottaa tietoa, on tilaaja (Subscriber). Tilaaja ja julkaisija eivät tiedä toistensa olemasta olosta, joten tarvitaan kolmas osapuoli. Kolmas osa kokonaisuudessa on MQTT Broker, joka tietää julkaisijan ja tilaajan. Näin broker eli välittäjä pystyy välittämään tiedot julkaisijalta tilaajalle. (HiveMQ, 2015)



Kuva 4 - MQTT:n toimintaperiaate

4.4 MySQL

MySQL on Oracle Corporationin omistama relaatiotietokantaohjelmisto. Se on käytetyimpiä avoimen lähdekoodin tietokantasovelluksia, ja sitä käyttävätkin useat tunnetut sivustot. Näitä ovat esimerkiksi Google ja Facebook. Tietokantana MySQL on yksinkertainen, toimintavarma ja korkean suorituskyvyn omaava. MySQL on saatavilla useimmille alustoille, mukaan lukien Linux, Mac OS ja Windows. Tietokannan hallinta tapahtuu joko komentorivillä, tai graafisella käyttöliittymällä kuten phpMyAdminilla. (MySQL, 2017)

5 IOT DATALOGGER TOTEUTUS

Toteutus vaatii kokonaisvaltaista tietotekniikan ymmärtämistä. Siihen liittyy verkon toimintaa, tietokannan toteutusta, Arduino-laitteen ohjelmointia sekä kytkentöjä. Projektin toteuttamiseksi tulee ottaa selvää eri toimintatavoista, ja selvittää mikä sopii kuhunkin kohtaan parhaiten.

5.1 Tietokannan rakentaminen - MySql

Toteutin tietokannan käyttäen phpMyAdmin MySQL-tietokannan hallintatyökalua. Aluksi loin tietokannan nimeltä temp_db. Tietokantaan loin taulun nimeltä temp_table, johon lisäsin sarakkeet laite_id (int), lampotila (decimal) sekä aika (timestamp).

```
CREATE DATABASE temp_db;
CREATE TABLE temp_table (laite_id INT PRIMARY KEY, lampotila DECIMAL,
aika timestamp);
```

5.2 WEB

Kun olin toteuttanut tietokannan, piti miettiä, kuinka saan tiedot selkeästi esille. Tiedot tulee olla helposti luettavassa muodossa selaimessa, joten päätin tehdä taulukon, johon haetaan seuraavat tiedot tietokannasta: Laitteen ID, lämpötilatieto sekä aikaleima. Web-osuuden toteutin PHP-kielellä. Aluksi pitää ottaa yhteys tietokantaan, jonka jälkeen haluamani tiedot ovat haettavissa. Tietokantayhteyteen tarvitaan palvelimen osoite, käyttäjänimi, salasana sekä tietokannan nimi. Jos yhteydessä on ongelmia, tulee viesti ”Yhteyttä ei voida muodostaa” (Kuva 4, Kuva 5).

connect.php

```
<?php

$user = 'root';
$pass = '';
$db = 'temp_db';

$db = new mysqli('localhost', $user, $pass, $db) or die("Yhteyttä ei voi muodostaa");


?>
```

Kuva 5 - Connect.php

tempinfo.php — Muokattu

```
<?php

require_once('../connect.php');

$query = "SELECT laite_id, lampotila, aika FROM temp_table";

$response = @mysqli_query($db, $query);

if ($response){

       echo '<table align="left" cellspacing="5" cellpadding="8">

       <tr><td align="left"><b> Laite ID </b></td>
       <td align="left"><b> Lampotila </b></td>
       <td align="left"><b> Aika </b></td></tr>';

while($row = mysqli_fetch_array($response)){

echo '<tr><td align="left">' .

$row['laite_id'] . '</td><td align="left">' .
$row['lampotila'] . '</td><td align="left">' .
$row['aika'] . '</td><td align="left">';
echo '</tr>';

}

echo '</table>';

}

else {
echo " Database Error <br />";
echo mysqli_error($db);

}

mysqli_close($db);

?>
```

Kuva 6 - Tempinfo.php

5.3 Mosquitto MQTT-broker

Viestin välitys IoT-laitteesta tietokantaan toteutettiin käyttämällä Mosquitto MQTT-brokeria. Se on avoimen lähdekoodin viestinvälitysohjelma, joka on saatavilla yleisimmille alustoille. Se perustuu teoriaosuudessa läpi käymääni MQTT-viestintäprotokollaan. Mosquitton lataaminen ja asennus on hyvin yksinkertainen. Se on ladattavissa mosquitto.org –Internet sivuilta. Asentaminen tapahtuu eri käyttöjärjestelmillä tyypillisillä tavoilla.

5.4 Python

Rajapinta MQTT välityspalvelimen sekä tietokannan välille toteutettiin Pythonilla (Kuva 7). Ensimmäiseksi tulee saada yhteys tietokannan ja välityspalvelimen välille. Yhteyden jälkeen pitää saada tilattua (subscribe) tiedot ”outTopic”-otsikosta. Merkkijono, joka outTopicista tulee, on muodossa {"deviceID":X,"temp":X}. Raakan JSON merkkijonon parsimiseen käytin json- kirjastoa. Tämän myötä saadaan tiedot sijoitettua temp_table taulukkoon; laite_id- ja lampotila sarakkeisiin.

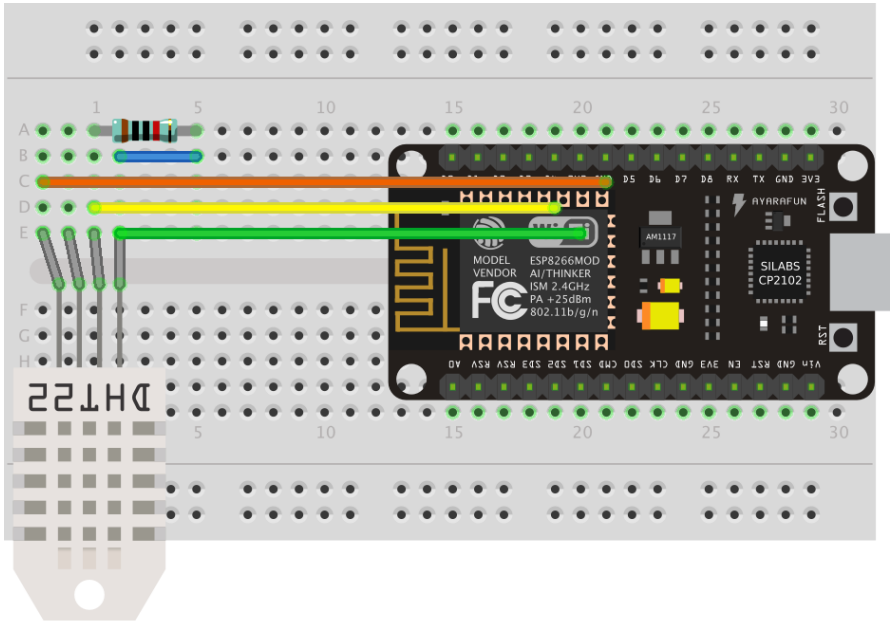
testConnect.py — Muokattu

```
import paho.mqtt.client as mqtt
import time
import json
import mysql.connector

cnx = mysql.connector.connect(user='root', password='',
                              host='127.0.0.1',
                              database='temp_db')
cursor = cnx.cursor()


def on_connect(client, userdata, flags, rc):
    m="Connected flags"+str(flags)+"result code "\
    +str(rc)+"client1_id  "+str(client)
    print(m)

def on_message(client1, userdata, message):
    if message.payload[0]=='{' and message.payload[-1]=='}':
        add_temp = "INSERT INTO temp_table (laite_id, lampotila) VALUES (%s, %s)"
        jsonArray = json.loads(str(message.payload))
        data_temp = (jsonArray['deviceID'], jsonArray['temp'])
        cursor.execute(add_temp, data_temp)
        cnx.commit()

broker_address="localhost"

client1 = mqtt.Client("serveri")
client1.on_connect= on_connect
client1.on_message=on_message
client1.connect(broker_address)

client1.subscribe("outTopic")

client1.loop_forever()


cursor.close()

cnx.close()
```

Kuva 7 - testConnect.py

5.5 Anturin kytkentä

DHT22-anturi kytkettiin NodeMcu:hun seuraavasti: VCC – 3V, DATA – Vastus (10k Ω 1%) –D4, GND – GND. Kytkennästä toteutin kytkentäkaavion Fritzing-ohjelmalla (Kuva 8).



Kuva 8 - Kytkentäkaavio

5.6 Alustan ohjelmointi

Alustan ohjelmointi tapahtui käyttäen Arduino IDE ohjelmointiympäristöä, jossa on saatavilla hyvin paljon esimerkkiohjelmia ja kirjastoja eri vaiheisiin. Alustan ohjelmointi koostuikin toimintojen yhdistelemisestä. Asian yksinkertaistamiseksi jaan sen vaiheisiin, jotka ovat seuraavat:

1. Yhteys
2. Anturin lukeminen
3. Halutun merkkijonon lähettäminen MQTT-brokerille

Alustaa ohjelmoitaessa tuli tietää selkeästi mitä haluan sen tekevän. Tavoitteena siis oli, että laite lähettää MQTT-brokerille merkkijonon, joka on muodossa {"deviceID":X,"temp":X}. Kaiken tämän mahdollistaakseen laitteen piti saada verkkoyhteys, joten se muodostetaan ensimmäisenä.

Yhteyden muodostamiseksi tarvitaan käyttöön ESP8266Wifi kirjasto. Verkkoyhteyden muodostamiseen tarvitaan myös langattoman verkon tiedot (verkon nimi ja salasana). Kun verkkoyhteys on muodostettu, tarvitaan yhteys myös MQTT-brokeriin. Siihen käytän PubSubClient kirjastoa, joka mahdollistaa myös viestin julkaisemisen (publish) MQTT palvelimelle. MQTT-broker yhteyteen tarvitaan myös palvelimen IP-osoite, joka on tässä tapauksessa sen tietokoneen IP, jolla alustaa ohjelmoidaan.

Kun yhteys on muodostettu, ja viesti kulkee laitteelta välityspalvelimelle, pitää saada laite lukemaan anturia. Käytin anturin lukemiseen DHT-kirjastoa. Anturin lukeminen vaatii lämpötilatietoa tuovan väylän, ja DHT anturin mallin. Anturin malli on DHT22, kuten teoriaosuudessa käytiin läpi. Koska anturin DATA on kytketty PIN D4:ään, on tietoa kuljettava väylä silloin D4.

NodeMcu on yhteydessä MQTT-brokeriin, ja sinne tulee eteenpäin välitettävä lämpötilatieto. Seuraavana tulee muodostaa JSON merkkijono, joka on muodossa {"deviceID":X,"temp":X}. Koska lämpötilatieto tulee Float-muodossa, on se muutettava String-muotoon, että se voidaan sisällyttää merkkijonoon. JSON merkkijono muodostui lopulta seuraavasti:

```
tempJSON = "{\"deviceID\":" + devId + ",\"temp\":" + t+ "}";
```

Koodissa “devID” sekä “t” ovat muuttujia. Laitteen id sisältyy ”devID”-muuttujaan, ja lämpötila sisältyy ”t”-muuttujaan.

6 JOHTOPÄÄTÖKSET

Tämän opinnäytetyön toteuttaminen auttoi ymmärtämään laajemmin tietoteknillisiä kokonaisuuksia. Esimerkiksi moniin ohjelmointikieliin tutustuin tarkemmin, ja sitä kautta opin uusia tapoja toimia. Välillä oli myös haasteita, sillä monet aihealueet ei olleet kovinkaan tuttuja minulle ennestään. Suurin haaste oli kuitenkin alkuun pääseminen. Kun olin päässyt alkuun, projekti tempaisi aika vahvasti mukaansa. Toteutusosuus eteni koko ajan nopeammin, sillä opin toimintatavan, miten etsiä tietoa sitä tarvittaessa.

Projektin toteuttaminen kesti noin kaksi kuukautta, ja kirjoitusosuus noin kuukauden. Pysyin aikataulussa, sillä projektin tuli olla valmiina vuoden vaihteeseen mennessä.

Toteutuksen aikana ilmeni monia kehitysideoita, joita projektiin voisi liittää. Tekstiviestillä ilmoittava varoitin, joka ilmoittaa lämpötilan laskiessa tietyn pisteen alapuolelle, olisi hyvä lisä. Lisäksi web-osuuteen voisi toteuttaa monenlaisia graafeja/diagrammeja, jolloin olisi selkeämpi seurata lämpötilan kehitystä.

LÄHTEET

Adafruit. 2017. Open-source hardware company. DHT22-anturi. Viitattu 28.1.2017 https://www.adafruit.com/product/385

Arduino. 2017. Arduinon virallinen kotisivu. Viitattu 23.1.2017 http://www.arduino.cc/

Ashton, K. 2009. That Internet of Things Thing. RFID Journal. Viitattu 18.1.2017 http://www.rfidjournal.com/articles/view?4986

Banzi, M. 2011. Arduino perusteista hallintaan. O'Reilly.

Benchoff, B. 2014. New Chip Alert: The ESP8266 WiFi Module (It's $5). Viitattu 31.1.2017 http://hackaday.com/2014/08/26/new-chip-alert-the-esp8266-wifi-module-its-5

Chan, H. C. 2015. Internet of Things Business Models. Journal of Service Science and Management. http://file.scirp.org/pdf/JSSM_2015081214471082.pdf

Hietanen, P. 2008. C++ ja olio-ohjelmointi. Docendo Finland Oy.

HiveMQ. 2015. MQTT Essentials: Part 1 – Introducing MQTT. Viitattu 30.1.2017 http://www.hivemq.com/blog/mqtt-essentials-part-1-introducing-mqtt

HiveMQ. 2015. MQTT Essentials: Part 2 – Publish & Subscribe. Viitattu 30.1.2017 http://www.hivemq.com/blog/mqtt-essentials-part2-publish-subscribe

Hovi, A. 2004. SQL-opas. Docendo Finland Oy. Sanoma WSOY konserni.

Laaksonen, A. 2011. PHP-ohjelmointi: Osa 1 – Johdanto. Ohjelmointiputka. Viitattu 26.1.2017. http://www.ohjelmointiputka.net/oppaat/opas.php?tunnus=php_01

MQTT. 2014. MQTT used by Facebook Messenger. Official website of MQTT. Viitattu 17.1.2017 http://mqtt.org/2011/08/mqtt-used-by-facebook-messenger

MQTT. 2014. Frequently asked questions. MQTT:n virallinen kotisivu. Viitattu 17.1.2017 http://www.mqtt.org/faq

MySQL. 2017. What is MySQL? MySQL:n virallinen kotisivu. Viitattu 28.1.2017. https://dev.mysql.com/doc/refman/5.7/en/what-is-mysql.html

Norris, D. 2015. The Internet of Things. McGraw-Hill Professional.

Python. 2017. General Python. Pythonin virallinen kotisivu. Viitattu 26.1.2017. http://www.python.org/

Rasila, A. 2004. Ohjelmoinnin alkeita Python-kielellä. Matematiikan ja tilastotieteen laitos. Helsingin yliopisto Viitattu 23.1.2017 http://matematiikkalehti-solmu.fi/2004/1/python.pdf