

Prototyypin raportti

Rakennemittari

versio 1.0

|  |  |  |
| --- | --- | --- |
| Karelia AMK | Tieto- ja viestintätekniikka | LTT6045 Suunnitteluprojekti |
| Tekijä: Jaakko Räsänen 1500911 | | Tulostettu: 8.5.2017 |
| Jakelu: Teemu Siponen 1500902, Jarke Koljonen 1501509, Mika Nissinen 1401304, Jesse Heiskanen 1401295 | | |
| Opettajat: Antti Rantaeskola, Ilpo Räsänen, Eero Väisänen | | |
|  | | |
|  | | |
| Dokumentin tila: Valmis | | Muokattu: 8.5.2017 |

Versiohistoria

|  |  |  |  |
| --- | --- | --- | --- |
| Versio | Päiväys | Tekijät | Selite (muutokset, korjaukset...) |
| 0.1 | 2.5.2017 | Jaakko Räsänen  Jarke Koljonen | Dokumentin aloitus |
|  |  |  |  |
| 1.0 | 8.5.2017 | Jaakko Räsänen  Jarke Koljonen  Teemu Siponen | Dokumentin kirjoittaminen  Dokumentin viimeistely |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Sisällysluettelo

1. JOHDANTO 4

1.1 Tarkoitus ja kattavuus 4

1.2 Määritelmät, termit ja lyhenteet 4

1.3 Yleiskatsaus dokumenttiin 4

2. Järjestelmän YLEISKUVAUS 5

3. Arkkitehtuurin kuvaukset Ja rajapinnat 6

3.1 Ratkaisuperiaatteet 6

3.2 Tietokanta-arkkitehtuuri 6

3.3 Moduulit ja prosessit 7

3.4 Fyysiset rajapinnat 8

3.5 Ohjelmistorajapinnat 9

4. Moduuli ja prosessi kuvaukset 10

4.1 Arduino-pohjainen mittari 10

4.2 Raspberry PI 3 11

4.3 ”Palvelinkone” 12

4.4 Dropbox - pilvipalvelu 13

5. Siirrettävyys 14

6. Ratkaisun rajoitteet 15

7. Testattavuus 16

8. Jatkokehitys 17

9. Liiteet 18

9.1 Arduino 19

9.2 Raspberry 20

9.3 Excel VBA 22

# JOHDANTO

## Tarkoitus ja kattavuus

Tämän dokumentin tarkoitus on kuvata ja selventää projektin aikana luodun prototyypin toimintaa. Dokumentissa on kuvatta sekä fyysiset että ohjelmalliset osa-alueet.

## Määritelmät, termit ja lyhenteet

|  |  |
| --- | --- |
| TERMI | KUVAUS |
| Arduino Nano | Datan mittaamiseen käytettävä mikrotietokone |
| Raspberry PI 3 | Datan tallentamiseen ja käsittelyyn käytettävä korttitietokone |
| Python | Ohjelmointikieli, joka sopii varsinkin matemaattisiin sovelluksiin. |
| NRF24L01+ | Langaton lähetin, jota käytetään datan siirtämiseen mittarilta. |
| Arduino IDE | Arduinon ohjelmointiin käytettävä ohjelma |
| SQL | Tietokanta, johon data tallennetaan |
| DHT22 (AM2302) | Kosteus- ja lämpötilamittari |

## Yleiskatsaus dokumenttiin

Dokumentti käsittelee projektin aikana luotua prototyyppiä. Dokumentin kappaleissa käydään lävitse prototyypin teknisiä ratkaisuja ja rajoitteita. Lopussa on esitetty mahdollisia jatkokehitysideioita.

# Järjestelmän YLEISKUVAUS

Mittarin (Arduino Nano) anturi (DHT22) mittaa rakenteiden lämpötilaa ja kosteutta. Mittari käyttää virtalähteenä 9 voltin litium paristoa. Mittarin mittaamat arvot lähetetään langattomasti korttitietokoneelle (Raspberry Pi 3). Korttitietokoneet ovat liitettynä verkkovirtaan ja ovat yhteydessä palvelimeen Ethernet kaapeleilla. Palvelinkone (Windows 7 tai uudempi) hakee tulokset korttitietokoneelta ja lähettää ne pilvi palvelimelle (Dropbox). Palvelinkoneella on kaksi verkkokorttia (Sisä- ja ulkoverkolle).

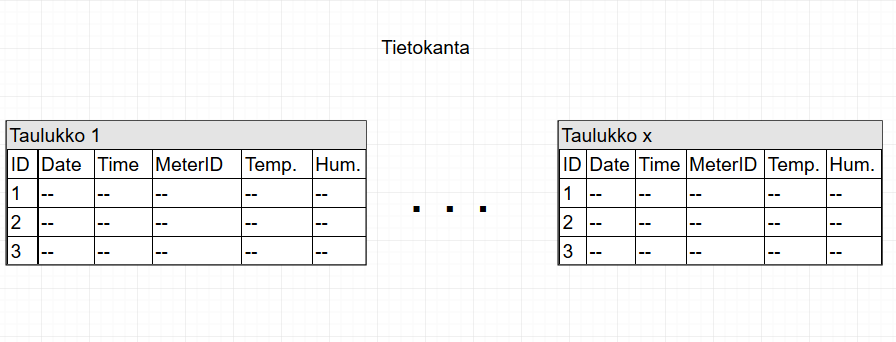
# Arkkitehtuurin kuvaukset Ja rajapinnat

## Ratkaisuperiaatteet

Teknisiä ratkaisuja tehtäessä on otettu huomioon koulun tarjoamat rajalliset komponentti vaihtoehdot.

## Tietokanta-arkkitehtuuri

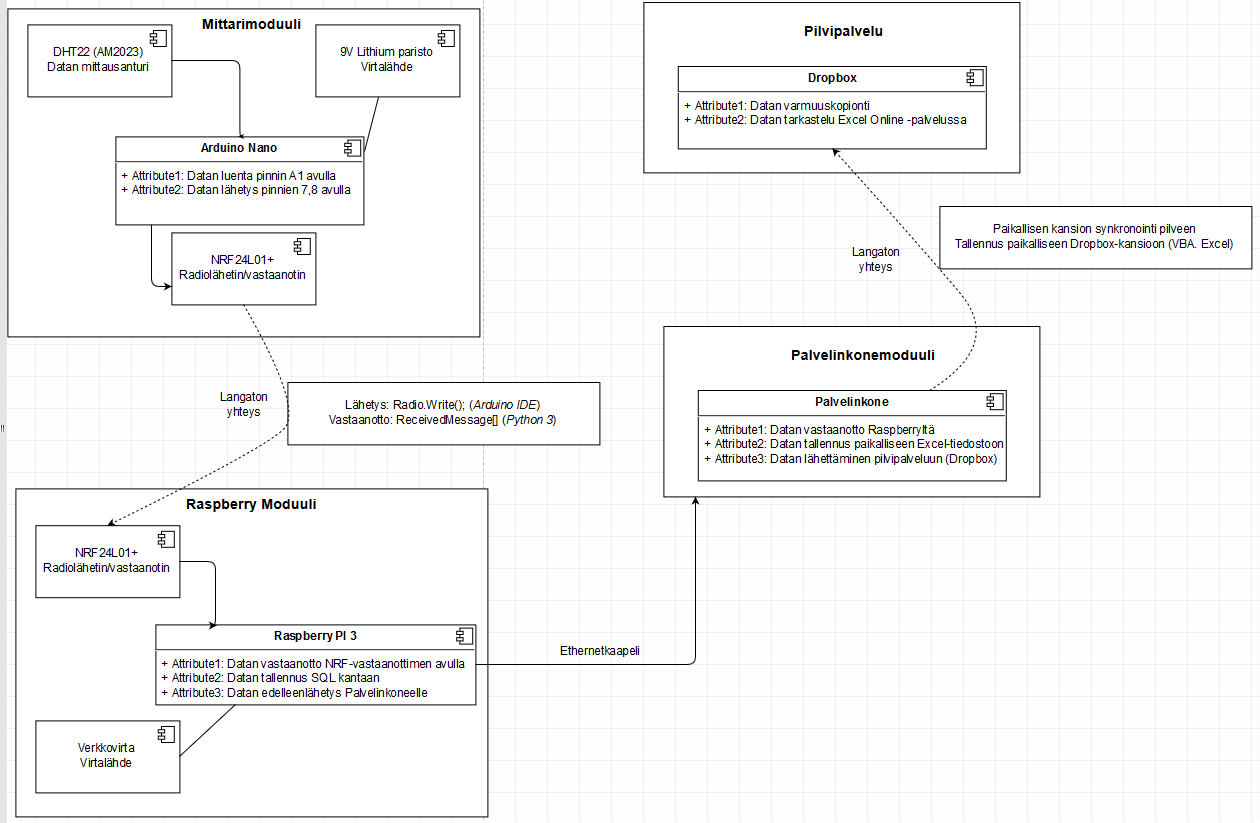
Alkuperäinen tallennus tehdään korttitietokoneilla MySQL tietokantaan. Palvelinkoneelle data muutetaan Excel tietokanta muotoon.



Kuva 1, Tietokannan taulukko esimerkki

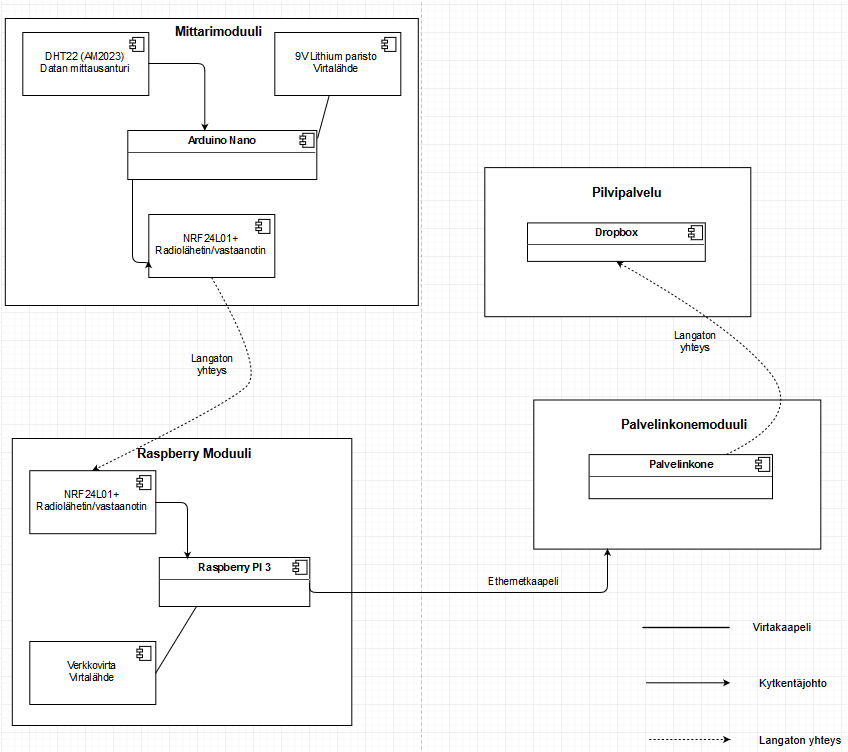
## Moduulit ja prosessit

Alla on kuvattu prototyyppiin luodut moduulit ja niiden prosessit. Kuvauksesta käy myös ilmi datan virtaussuunnat järjestelmän sisällä.



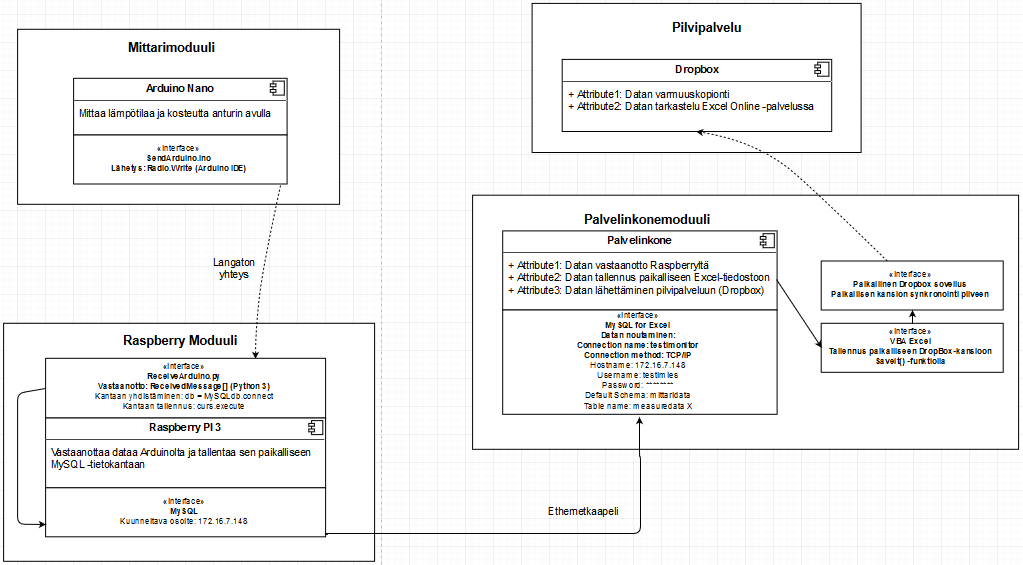
Kuva 2, Moduulit ja prosessit

## Fyysiset rajapinnat



Kuva 3, Fyysiset rajapinnat

## Ohjelmistorajapinnat



Kuva 4, Ohjelmistorajapinnat

# Moduuli ja prosessi kuvaukset

Seuraava osio selittää tarkemmin, kuinka järjestelmän eri moduulit toimivat. Tällä sivulla selitetyt prosessi ja toiminnot löytyvät koodi muodossa Prototyyppi dokumentin liitteistä 1 ja 2.

## Arduino-pohjainen mittari

|  |  |
| --- | --- |
| KUVAUS | SELITE |
| Yleiskuvaus | Arduino piiriin liitetty mittari mittaa kosteutta ja lämpötilaa ja lähettää sen Raspberry PI korttitietokoneelle NRF24L01+ lähettimen avulla. |
| Attribuutit | dht.readHumidity()  dht.readTemperature()  ID  testi[i] |
| Operaatiot | Aluksi mittariin alustetaan käytettävät kirjastot ja määritetään mittaukseen ja lähettämiseen käytettävät pinnit.  Tämän jälkeen asetetaan kertaluontoisesti seuraavat kohdat:   * Sarjaportti ja sen bittinopeus * Lähettimen:   + Virtataso   + Kanava   + Putki   + Dynaaminen koko   + Käynnistys * Käynnistetään mittari   Silmukassa pyöritetään kolmea muuttujaa, jotka tallennetaan lopuksi joukko muuttujaan. Muuttujiin tallennetaan mittaustulokset ja mittarin ID. Kun mittaustulokset on saatu, tallennetaan muuttujat joukkoon, joka lähetetään Raspberry PI:lle purettavaksi. |
| Poikkeus- ja virhetilanteiden käsittely | Poikkeus- ja virhetilanteita ei käsitellä vielä automaattisesti. |

## Raspberry PI 3

|  |  |
| --- | --- |
| KUVAUS | SELITE |
| Yleiskuvaus | Raspberry PI 3 korttitietokone vastaanottaa Arduino mittarimoduulin lähettämää dataa ja tallentaa sen MySQL tietokantaan jatkokäsittelyä varten. |
| Attribuutit | receivedMessage[]  MeterID  Humidity  Temperature |
| Operaatiot | Aluksi korttitietokoneeseen alustetaan käytettävät kirjastot ja tiedostot.  Tämän jälkeen asetetaan kertaluontoisesti seuraavat kohdat:   * Tietokantayhteys * Vastaanottimen:   + Putket   + Pinnit   + Vastaanotetun tiedoston maksimikoko   + Datanopeus   + Virtataso   + Automaattinen kuittaus   + Dynaamisten kokojen käyttöönotto   + Niiden kuittaus   + Putkien aukaisu   + Kuuntelun aloitus   Silmukka kuuntelee vastaanottimeen tulevia viestejä. Kun viesti saapuu, se tallennetaan joukko muuttujaan. Tästä joukosta puretaan tietyt alkiot ja asetetaan ne tiettyjen muuttujien arvoiksi. Nämä muuttujat muunnetaan JSON merkkijonoiksi. Muuttujat tallennetaan sen jälkeen MySQL- tietokantaan. |
| Poikkeus- ja virhetilanteiden käsittely | Poikkeus- ja virhetilanteita ei käsitellä vielä automaattisesti. |

## ”Palvelinkone”

|  |  |
| --- | --- |
| KUVAUS | SELITE |
| Yleiskuvaus | Päätekoneelle asennettu Excel ohjelma noutaa MySQL muotoista dataa siihen yhteydessä olevilta Raspberry PI korttitietokoneilta. Tähän datan hakemiseen hyödynnetään MySQL for Excel lisäosaa.  Tämän jälkeen Excel luo VBA moduulin avulla pyörivästä tiedostosta kopion ja tallentaa sen paikalliseen Dropbox- kansioon. Tämä kansio päivittyy automaattisesti pilvipalveluun. |
| Operaatiot | MySQL for Excel lisäosa mahdollistaa yhteyden luomisen ulkopuolisiin MySQL lähteisiin. Tämä yhteyden pystytys on kuvattu Prototyyppi dokumentissa.  MySQL for Excel hakee dataa automaattisesti tietyllä aikavälillä korttitietokoneiden MySQL- tietokannoista. Jokainen yhteys täytyy pystyttää kerran, jonka jälkeen järjestelmä toimii automaattisesti seuraavaan virhe- tai poikkeustilanteeseen asti. Exceliin on tehty VBA moduuli jonka Saveit() funktio tallentaa automaattisesti käynnissä olevan tiedoston ja luo siitä varmuuskopion. Tämä varmuuskopio tallennetaan paikalliseen Dropbox- kansioon, josta se lähetetään Dropbox- pilvipalveluun. |
| Poikkeus- ja virhetilanteiden käsittely | Poikkeus- ja virhetilanteita ei käsitellä vielä automaattisesti. |

## Dropbox - pilvipalvelu

|  |  |
| --- | --- |
| KUVAUS | SELITE |
| Yleiskuvaus | Excelin tallentama tiedosto synkronoituu automaattisesti Dropboxin palvelimelle. Tälle palvelimelle on varattu 2GB tallennustilaa.  Palvelimelle tallennettua dataa pystyy tarkastelemaan kaikki jotka tietävät kirjautumiseen tarvittavat tunnukset tai joille tämä data tiedosto on jaettu. Datan tarkastelu tapahtuu Excel Online ohjelman avulla, joka on sisään rakennettuna Dropbox palveluun. |
| Poikkeus- ja virhetilanteiden käsittely | Poikkeus- ja virhetilanteita ei voida käsitellä automaattisesti. Palvelu on täysin Dropbox Oy hallinnassa |

# Siirrettävyys

Prototyyppi on pieni ja helposti mukana kannettava paketti.

Vain pieni osa valmiista tuotteista on siirrettävissä asennuksen jälkeen niiden sijoittelun vuoksi.

# Ratkaisun rajoitteet

Tuotteet ovat toimivia vain niiden patterien keston ajan. Tuotteiden sijoittelun vuoksi niitä ei voidata ladata pariston loppumisen jälkeen. Vain pieni osa tuotteista on kerättävissä ja lataamisen jälkeen uudelleen käytettävissä.

# Testattavuus

Prototyypin testaaminen onnistuu vain osittain paremman testiympäristön uupumisen vuoksi.

Prototyypin kantoalueen testaaminen (tämäkään ei aivan täydellisesti) ja virrankulutus ovat ainoat testit jotka voidaan suorittaa ilman sopivaa testiympäristöä.

# Jatkokehitys

* 9V pariston voisi vaihtaa 3.7V litium paristoon, jolloin Arduinon rekulaattorin voisi poistaa. Tällä tavalla pariston kestoa voidaan pidentää huomattavasti.
* Arduinoon tulisi lisätä ”sleepmode”, jolloin Arduinon virrankulutusta voitaisiin pienentää.
* Prototyypissä esitetyn palvelintietokoneen voi hävittää välistä, mikäli sen ei katsota sopivan lopulliseen ratkaisuun. Tällöin Raspberryt tai yksi välitys Raspberry on yhteydessä Internettiin ja lataa tallennetun datan CSV-tiedostomuodossa verkkoon. Tällöin tarkasteltavan tiedoston ulkonäön muokkaaminen jää kuitenkin käyttäjän vastuulle.
* Jos palvelinkone päätetään jättää pois, tulee Internettiin yhteydessä olevien Raspberryihin asentaa vähintään palomuuri.

# Liiteet

1. Arduinon koodi
2. Raspberry PI 3 koodi
3. Excel VBA Module koodi

## Arduino

#include<DHT.h>

#include<SPI.h>

#include<RF24.h>

#include<rBase64.h>

#define DHTPIN A1

#define DHTTYPE DHT22

DHT dht(DHTPIN, DHTTYPE);

RF24 radio(7,8);

void setup(void)

{

Serial.begin(9600);

radio.begin();

radio.setPALevel(RF24\_PA\_MAX);

radio.setChannel(0x76);

radio.openWritingPipe(0xF0F0F0F0E1LL);

radio.enableDynamicPayloads();

radio.powerUp();

dht.begin();

}

void loop(void)

{

const uint8\_t hum = dht.readHumidity();

const uint8\_t temp = dht.readTemperature();

const uint8\_t ID = 1;

int testi[3] = {ID, hum, temp};

int size\_of\_testi = sizeof(testi);

Serial.println(testi[0]);

Serial.println(testi[1]);

Serial.println(testi[2]);

radio.write(&testi, sizeof(testi));

delay(2000);

/\*

Serial.println(hum);

radio.write(&hum, sizeof(hum));

delay(2000);

Serial.println(temp);

radio.write(&temp, sizeof(temp));

delay(1000);

\*/

// DISPLAY DATA

/\*Serial.println("Humidity is: ");

Serial.println(hum);

Serial.println("Temperature is: ");

Serial.println(temp);

\*/

delay(1000);

}

## Raspberry

#!/usr/bin/env python

#!/usr/bin/python

import demjson

import pymysql

pymysql.install\_as\_MySQLdb()

import MySQLdb

db = MySQLdb.connect("172.16.7.120", "jessetestaa", "password", "mittaridata")

curs=db.cursor()

import RPi.GPIO as GPIO

from lib\_nrf24 import NRF24

import time

import spidev

GPIO.setmode(GPIO.BCM)

pipes = [[0xE8, 0xE8, 0xF0, 0xF0, 0xE1], [0xF0, 0xF0, 0xF0, 0xF0, 0xE1]]

radio = NRF24(GPIO, spidev.SpiDev())

radio.begin(0, 17)

radio.setPayloadSize(32)

radio.setChannel(0x76)

radio.setDataRate(NRF24.BR\_1MBPS)

radio.setPALevel(NRF24.PA\_MIN)

radio.setAutoAck(True)

radio.enableDynamicPayloads()

radio.enableAckPayload()

radio.openReadingPipe(1, pipes[1])

radio.printDetails()

radio.startListening()

while(1):

# ackPL = [1]

while not radio.available(0):

print("Tämä on viestin tarkistus looppi")

time.sleep(1 / 1)

receivedMessage = []

radio.read(receivedMessage, radio.getDynamicPayloadSize())

print("Received: {}".format(receivedMessage))

print("tarkastus")

MeterID = receivedMessage[0]

Humidity = receivedMessage[2]

Temperature = receivedMessage[4]

#for i in range(len(receivedMessage)):

# print(i,receivedMessage[i]);

print(MeterID)

print(Humidity)

print(Temperature)

json = demjson.encode(Temperature)

json2 = demjson.encode(MeterID)

json3 = demjson.encode(Humidity)

with db:

curs.execute ("INSERT INTO measuredata (tdate, ttime, meterID, humidity, temperature) VALUES(CURRENT\_DATE(), NOW(), "+str(MeterID)+", "+str(Humidity)+", "+str(Temperature)+" )")

db.commit()

print("data committed")

## Excel VBA

Sub SaveIt()

ThisWorkbook.Save

Application.Wait (Now + TimeSerial(0, 0, 30))

ThisWorkbook.SaveCopyAs "C:\Users\1500911\Dropbox\mittaridatabackup.xlsm"

Application.OnTime Now + TimeSerial(0, 0, 50), "SaveIt"

End Sub