

ALKUUN ▾

HARRASTEKANALA

OPIKSELU

REMONTIT

SUKUTUTKIMUS

TEE SE ITSE

YLEINEN

VALIKKO



VALIKKO



(C) Jari Hiltunen



(C) Jari Hiltunen

Lämpötila- ja kosteusarvodatojen luku ja mqtt-protokollan käyttöönotto

□ DIVERGENT □ 26.5.2020 □ HARRASTEKANALA

Kanalan automaatioprojektissa ([bloggaus tässä](#)) helpointa on aloittaa DHT-22 lämpötila- ja kosteusantureista. Projektista tulee oletettavasti aika iso ja koodia tulee paljon. Tällöin sovelluksen suunnittelu pitää lähteä siitä, että käytetään olioita ja tehdään niihin mahdollisimman monikäyttöisiä luokkia metodeineen.

Python-ohjelmoinnin kannalta antureille voisi siis tehdä oman luokkansa (olion), jossa olisi kerrottuna mitä gpio-pinniä luetaan, mutta tämä ei vaikuta olevan järkevää prosessoritehon kannalta eikä oikeastaan edes de-facto-ohjelmointitapanakaan. Konstruktorissa ajattelin esitellä siis anturin nimen, tarkoituksen, pinnin ja sitten listata eri metodeja, joilla eri antureita käsiteltäisiin ja niiltä saatuja arvoja palautettaisiin. Tällöin joutuisi käytännössä ajamaan

VIIMEISIMMÄT ARTIKKELIT

[InfluxDB-
tietokannan ja
Grafanan asennus
Raspberryyn](#)

27.5.2020

[Lämpötila- ja
kosteusarvodatojen
luku ja mqtt-
protokollan
käyttöönotto](#)

26.5.2020

[Ryobi RY18PCB-
140 puhdistusharja](#)

19.5.2020

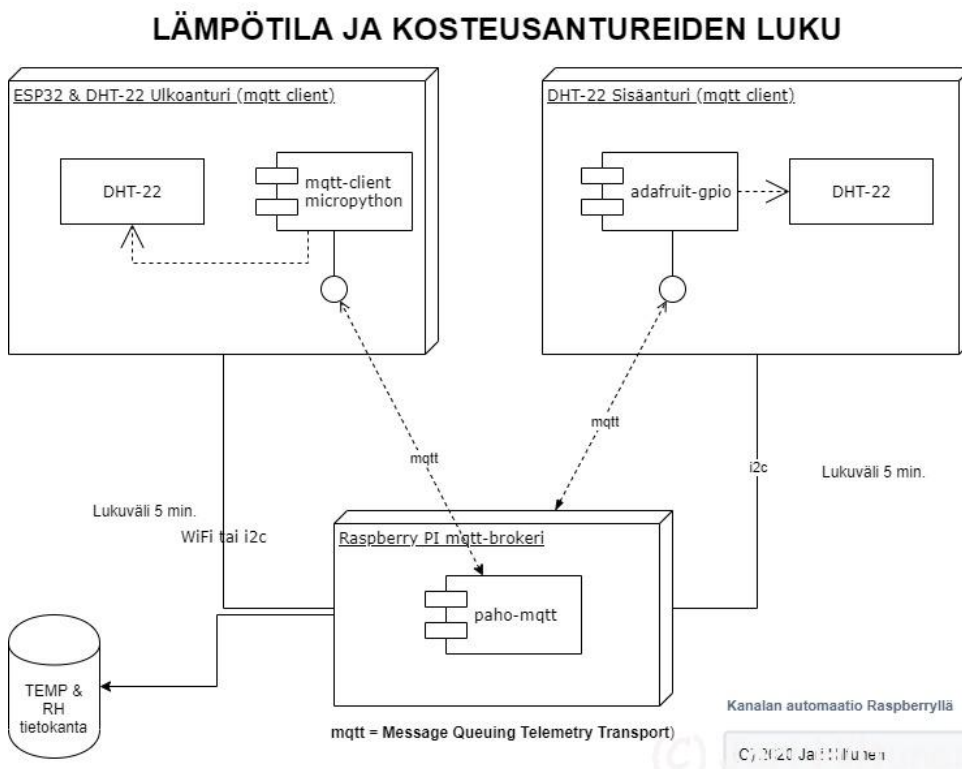
sovellusta thredeillä (threading) ja tämä söisi resursseja ja olisi vaikeasti hallittavissa.

Koska threadit eivät oikein tule kyseeseen, viisaampi lähtökohta on hyödyntää Internet of Things (iot), eli [esineiden internetiä](#), jossa laitteet lähettelevät viestejä eri yhteysmetodien avulla.

Raspberryn maailmassa IoT tarkoittaa kevyimmillään sitä, että hyödynnetään [mqtt-protokollaa](#). MQTT:ssä on oikeastaan vain kaksi tärkeää asiaa, aiheet ja viestit. Julkaisijat (publishers) lähettävät viestejä aiheiden mukaan. Aiheet ovat ikäänkuin kanavia.

Esimerkiksi julkaisija (publisher, clientti) voi lähettää viestin temp: 20.2 aiheena lämmitys/sensori/kanala. Tilaaajat (subscriber) kuuntelevat aiheviestejä ja toteuttavat viestin saatuaan mitä vain. Eli kaikki tilaajat, jotka kuuntelevat aihetta lämmitys/sensori/kanala saavat viestin temp: 20.2. Välittäjä (serveri, broker) välittää aiheet ja viestit tilaajien ja julkaisijoiden välillä.

Ajatuskaaviona lämpötilan- ja kosteusarvon luku tapahtuisi siis kutakuinkin näin:



Hyvä puoli toteutuksessa on se, että järjestelmä voidaan hajauttaa miten halutaan, esimerkiksi [ESP32:lle](#) (NodeMCU Wifi+Bluetooth ominaisuuksilla maksaa noin 3 – 10€ riippuen mistä tilaa) voi tehdä clientin lähettämään DHT-22 luettavaa tietoa ja kyseisen moduulin

V380 Pro valvontakameran räjäytys

17.5.2020

Tee se itse
läpinäkyvä covid-
19 räkäsuoja tai
visiiri

15.5.2020

ARKISTOT

Valitse kuukau

AVAINSANAT

3d (11)

aikuiskasvatus (8)

covid19 (6)

DIY (24)

Etäohjaus (6)

genetiikka (8)

hanko (16)

harrastekanat (15)

huolto (6)

idea (7)

kanala (23)

kanat (23)

kehityopsykologia
(6)

keinoäly (7)

kesäkanat (13)

kiinalainen (6)

korjaus (10)

voi viedä miten kauaksi tahansa. Tällöin ei esimerkiksi kaapeloinnista synny ongelmia, kuten kaapeliin indusoituneet häiriöt tai kaapelin häviöistä johtuvat ongelmat. Valokuitu olisi toki eräs vaihtoehto häiriöetäisyyden suhteen.

ESP32 kuluttaa virtaa syvässä lepotilassa 7 uA, lepotilassa 1 mA ja normaalioperaatiossa (240 MHz) 50 mA ja WiFin kanssa 80 – 180 mA. Virtalähteeksi sopii vähän päällä oleviin sovellutuksiin parhaiten litium paristot, kuten CR123, jolloin syvässä lepotilassa ESP32 toimisi jopa 5 vuotta. ESP32:lle saa 18650 akkua hyödyntäviä valmiita moduleita, jotka maksavat noin 3 – 11€ riippuen miten monta akkua haluaa käyttää. Akut makavat noin 5€ kappale. Muuten esimerkiksi valokennolla latautuva akkupankki tai muu vastaava sopii hyvin, mutta tavallaan on turha käyttää yli 3,3V jännitteitä.

Mikään ei estä liittämästä i2c:llä Raspberryyn samaisia antureita, eli Raspi voi toimia sekä brokerina että clienttina.

DHT-22 kokemusten perusteella minimi lukuväli on 2 sekuntia, mutta järkevä lukuväli voisi olla esimerkiksi 5 minuuttia, sillä lämpötilan ja kosteusarvojen lukeminen tuskin on kovin kriittinen toiminta.

Aluksi mitatut arvot voisi tallentua csv-formaattiin tiedostoksi Raspberyllle. Myöhemmin mitatut arvot voisi tallentua esimerkiksi MySQL-palvelimeen, joka ei tarvitse olla Raspberry, vaan jossain muualla verkossa.

MQTT-brokerin ja clientin asennus Raspberryyn

Raspissa käytetään mosquitto-mqtt-brokeria. Sen saa asennettua komennolla `sudo apt install mosquitto mosquitto-clients`

koulutus (6)

käytös (6)

laki (7)

lääketiede (6)

lääkitys (7)

maatiaisana (20)

masennus (6)

neurologia (8)

opiskelu (22)

oppiminen (9)

otex (6)

paneelikattomaali (7)

pedagogia (6)

pelko (6)

pintaremontti (9)

ponttimaali (7)

psykologia (18)

pääsiäisparvi (19)

raspberry (6)

remontti (8)

stressi (7)

Suunnittelu (6)

teeseitse (8)

tikkurila (6)

tutkimus (9)

valvontakamera (7)

video (18)

videovalvonta (7)

```
pi@kanala-raspi: ~  
Tiedosto Muokkaa Väliohje Ohje  
pi@kanala-raspi:~ $ sudo apt install mosquitto mosquitto-clients  
Luetaan pakettiluetteloita... Valmis  
Muodostetaan riippuvuussuhteiden puu  
Luetaan tilatiedot... Valmis  
The following additional packages will be installed:  
  libev4 libmosquitto1 libwebsockets8  
Seuraavat UUDET paketit asennetaan:  
  libev4 libmosquitto1 libwebsockets8 mosquitto mosquitto-clients  
0 päivitetty, 5 uutta asennusta, 0 poistettavaa ja 0 päivittämätöntä.  
Noudettavaa arkistoa 388 kt.  
Toiminnon jälkeen käytetään 843 k t lisää levytilaa.  
Haluatko jatkaa? [K/e] ☐
```

(C) Jari Hiltunen

Aktivointi tapahtuu komennolla `sudo systemctl enable mosquitto` ja statuksen näkee komennolla `sudo systemctl status mosquitto`

```
pi@kanala-raspi: ~  
Tiedosto Muokkaa Väliohje Ohje  
Selecting previously unselected package libwebsockets8:armhf.  
Preparing to unpack .../libwebsockets8_2.0.3-3_armhf.deb ...  
Unpacking libwebsockets8:armhf (2.0.3-3) ...  
Selecting previously unselected package mosquitto.  
Preparing to unpack .../mosquitto_1.5.7-1+deb10u1_armhf.deb ...  
Unpacking mosquitto (1.5.7-1+deb10u1) ...  
Selecting previously unselected package mosquitto-clients.  
Preparing to unpack .../mosquitto-clients_1.5.7-1+deb10u1_armhf.deb ...  
Unpacking mosquitto-clients (1.5.7-1+deb10u1) ...  
Setting up libmosquitto1:armhf (1.5.7-1+deb10u1) ...  
Setting up libev4:armhf (1:4.25-1) ...  
Setting up mosquitto-clients (1.5.7-1+deb10u1) ...  
Setting up libwebsockets8:armhf (2.0.3-3) ...  
Setting up mosquitto (1.5.7-1+deb10u1) ...  
Created symlink /etc/systemd/system/multi-user.target.wants/mosquitto.service →  
/lib/systemd/system/mosquitto.service.  
Processing triggers for systemd (241-7~deb10u4+rp11) ...  
Processing triggers for man-db (2.8.5-2) ...  
Processing triggers for libc-bin (2.28-10+rp11) ...  
pi@kanala-raspi:~ $ sudo systemctl enable mosquitto  
Synchronizing state of mosquitto.service with SysV service script with /lib/syst  
emd/systemd-sysv-install.  
Executing: /lib/systemd/systemd-sysv-install enable mosquitto  
pi@kanala-raspi:~ $
```

(C) Jari Hiltunen


```
pi@kanala-raspi: ~
Tiedosto Muokkaa Vällilehdet Ohje
Created symlink /etc/systemd/system/multi-user.target.wants/mosquitto.service → /lib/systemd/system/mosquitto.service.
Processing triggers for systemd (241-7~deb10u4+rpil) ...
Processing triggers for man-db (2.8.5-2) ...
Processing triggers for libc-bin (2.28-10+rpil) ...
pi@kanala-raspi:~ $ sudo systemctl enable mosquitto
Synchronizing state of mosquitto.service with SysV service script with /lib/systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable mosquitto
pi@kanala-raspi:~ $ sudo systemctl status mosquitto
● mosquitto.service - Mosquitto MQTT v3.1/v3.1.1 Broker
   Loaded: loaded (/lib/systemd/system/mosquitto.service; enabled; vendor preset
   Active: active (running) since Mon 2020-05-25 11:08:29 EEST; 2min 24s ago
     Docs: man:mosquitto.conf(5)
           man:mosquitto(8)
   Main PID: 1118 (mosquitto)
     Tasks: 1 (limit: 2200)
    Memory: 816.0K
   CGroup: /system.slice/mosquitto.service
           └─1118 /usr/sbin/mosquitto -c /etc/mosquitto/mosquitto.conf

touko 25 11:08:29 kanala-raspi systemd[1]: Starting Mosquitto MQTT v3.1/v3.1.1 B
touko 25 11:08:29 kanala-raspi systemd[1]: Started Mosquitto MQTT v3.1/v3.1.1 Br
lines 1-13/13 (END)
```

Aihe on yksinkertaisesti merkkijono, jota kuunnellaan.

Merkinjonossa /, + ja # ovat varattuja, muuten merkkijono on vapaata riistaa.

Voit esimerkiksi tilata aiheen testi/viesti-viestit komennolla
mosquitto_sub -h localhost -t "testi/viesti"

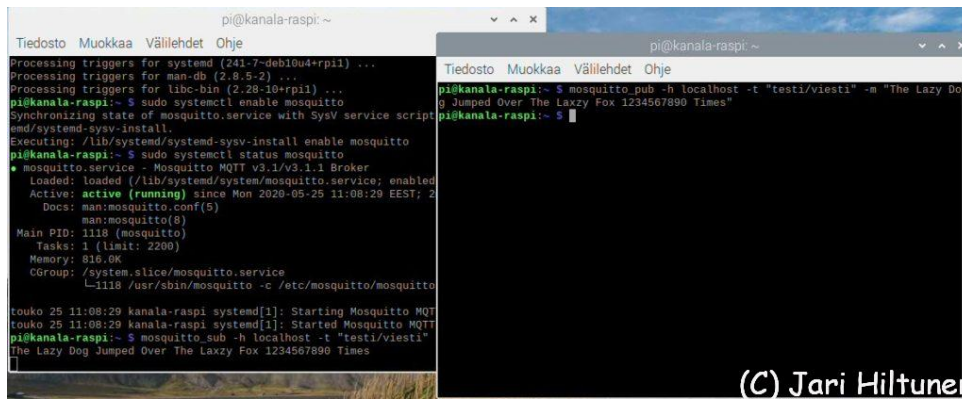
```
pi@kanala-raspi: ~
Tiedosto Muokkaa Vällilehdet Ohje
/lib/systemd/system/mosquitto.service.
Processing triggers for systemd (241-7~deb10u4+rpil) ...
Processing triggers for man-db (2.8.5-2) ...
Processing triggers for libc-bin (2.28-10+rpil) ...
pi@kanala-raspi:~ $ sudo systemctl enable mosquitto
Synchronizing state of mosquitto.service with SysV service script with /lib/systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable mosquitto
pi@kanala-raspi:~ $ sudo systemctl status mosquitto
● mosquitto.service - Mosquitto MQTT v3.1/v3.1.1 Broker
   Loaded: loaded (/lib/systemd/system/mosquitto.service; enabled; vendor preset
   Active: active (running) since Mon 2020-05-25 11:08:29 EEST; 2min 24s ago
     Docs: man:mosquitto.conf(5)
           man:mosquitto(8)
   Main PID: 1118 (mosquitto)
     Tasks: 1 (limit: 2200)
    Memory: 816.0K
   CGroup: /system.slice/mosquitto.service
           └─1118 /usr/sbin/mosquitto -c /etc/mosquitto/mosquitto.conf

touko 25 11:08:29 kanala-raspi systemd[1]: Starting Mosquitto MQTT v3.1/v3.1.1 B
touko 25 11:08:29 kanala-raspi systemd[1]: Started Mosquitto MQTT v3.1/v3.1.1 Br
pi@kanala-raspi:~ $ mosquitto_sub -h localhost -t "testi/viesti"
```

Komento lähettää tilauspyynnön brokerille (tässä tapauksessa localhost, eli raspi itsessään). Brokeri voi toki olla missä tahansa mikä tahansa kone. Kuten huomaat, jää terminaali odottamaan testi/viesti-aiheella lähetettyjä viestejä.

Pidetään terminaaliruutu auki ja avataan toinen terminaaliruutu, josta julkaistaan aiheena "testi/viesti" viesti "The Lazy Dog Jumped

Over The Lazy Fox 1234567890 Times” komennolla mosquitto_pub -h localhost -t ”testi/viesti” -m ”The Lazy Dog Jumped Over The Lazy Fox 1234567890 Times”

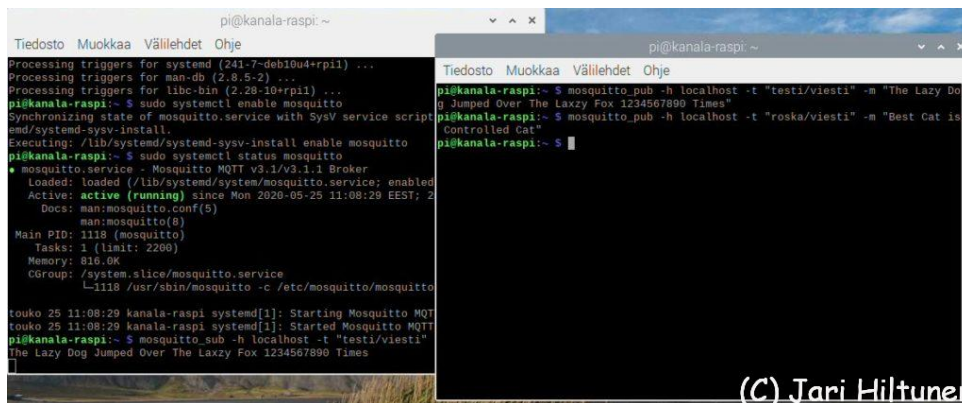


```
pi@kanala-raspi: ~  
Tiedosto Muokkaa Välllehdet Ohje  
Processing triggers for systemd (241-7-deb10u4+rp1) ...  
Processing triggers for man-db (2.8.5-2) ...  
Processing triggers for libc-bin (2.28-10+rp1) ...  
pi@kanala-raspi:~$ sudo systemctl enable mosquitto  
Synchronizing state of mosquitto.service with SysV service script  
emd/systemd-sysv-install.  
Executing: /lib/systemd/systemd-sysv-install enable mosquitto  
pi@kanala-raspi:~$ sudo systemctl status mosquitto  
● mosquitto.service - Mosquitto MQTT v3.1/v3.1.1 Broker  
   Loaded: loaded (/lib/systemd/system/mosquitto.service; enabled  
   Active: active (running) since Mon 2020-05-25 11:08:29 EEST; 2  
     Docs: man:mosquitto.conf(5)  
   Main PID: 1118 (mosquitto)  
     Tasks: 1 (limit: 2200)  
    Memory: 816.0K  
   CGroup: /system.slice/mosquitto.service  
           └─1118 /usr/sbin/mosquitto -c /etc/mosquitto/mosquitto  
touko 25 11:08:29 kanala-raspi systemd[1]: Starting Mosquitto MQTT  
touko 25 11:08:29 kanala-raspi systemd[1]: Started Mosquitto MQTT  
pi@kanala-raspi:~$ mosquitto_sub -h localhost -t "testi/viesti"  
The Lazy Dog Jumped Over The Lazy Fox 1234567890 Times  
pi@kanala-raspi:~$
```

(C) Jari Hiltunen

Näet toisella ruudulla vastaanotetun viestin (koska aihe, jota kuunnellaan testi/viesti nähtiin). Mikäli aiheena olisi ollut jotain muuta, ei viestiä olisi näkynyt.

Alla olevassa kuvassa viestin aiheena onkin roska/viesti, eikä sitä näytetä vasemmassa terminaaliruudussa:



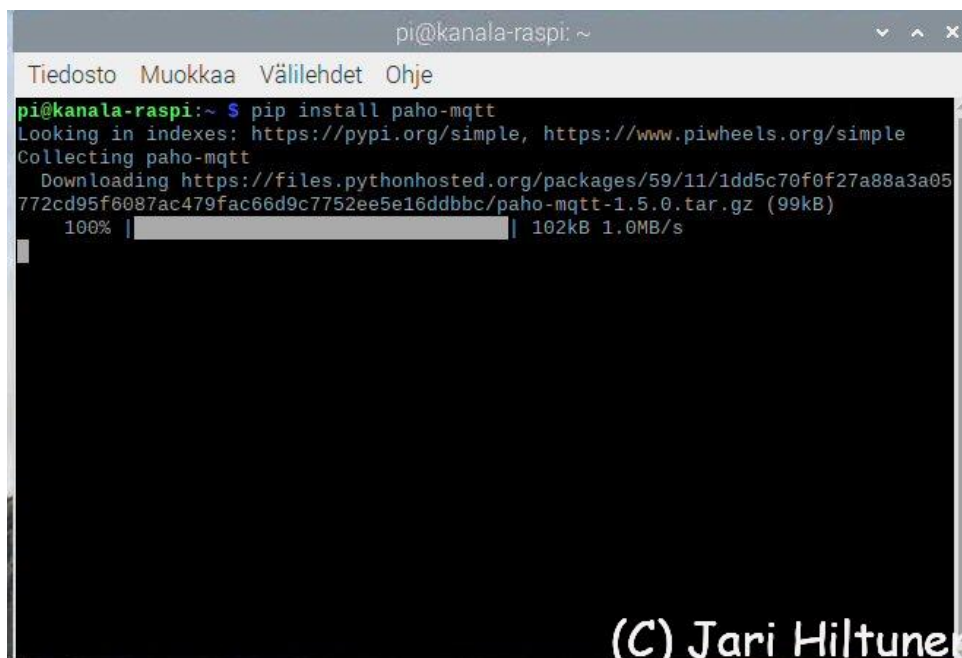
```
pi@kanala-raspi: ~  
Tiedosto Muokkaa Välllehdet Ohje  
Processing triggers for systemd (241-7-deb10u4+rp1) ...  
Processing triggers for man-db (2.8.5-2) ...  
Processing triggers for libc-bin (2.28-10+rp1) ...  
pi@kanala-raspi:~$ sudo systemctl enable mosquitto  
Synchronizing state of mosquitto.service with SysV service script  
emd/systemd-sysv-install.  
Executing: /lib/systemd/systemd-sysv-install enable mosquitto  
pi@kanala-raspi:~$ sudo systemctl status mosquitto  
● mosquitto.service - Mosquitto MQTT v3.1/v3.1.1 Broker  
   Loaded: loaded (/lib/systemd/system/mosquitto.service; enabled  
   Active: active (running) since Mon 2020-05-25 11:08:29 EEST; 2  
     Docs: man:mosquitto.conf(5)  
   Main PID: 1118 (mosquitto)  
     Tasks: 1 (limit: 2200)  
    Memory: 816.0K  
   CGroup: /system.slice/mosquitto.service  
           └─1118 /usr/sbin/mosquitto -c /etc/mosquitto/mosquitto  
touko 25 11:08:29 kanala-raspi systemd[1]: Starting Mosquitto MQTT  
touko 25 11:08:29 kanala-raspi systemd[1]: Started Mosquitto MQTT  
pi@kanala-raspi:~$ mosquitto_sub -h localhost -t "testi/viesti"  
The Lazy Dog Jumped Over The Lazy Fox 1234567890 Times  
pi@kanala-raspi:~$
```

(C) Jari Hiltunen

Tämä sama periaate toimii myös kotiautomaatiossa ja muissa sovellutuksissa.

Python ja mqtt-kirjasto

Jotta mqtt toimisi Pythonin kanssa, tulee asentaa Paho-MQTT-kirjasto. Asennus tapahtuu komennolla pip install paho-mqtt



```
pi@kanala-raspi: ~  
Tiedosto Muokkaa Välilehdet Ohje  
pi@kanala-raspi:~$ pip install paho-mqtt  
Looking in indexes: https://pypi.org/simple, https://www.piwheels.org/simple  
Collecting paho-mqtt  
  Downloading https://files.pythonhosted.org/packages/59/11/1dd5c70f0f27a88a3a05772cd95f6087ac479fac66d9c7752ee5e16ddbcb/paho-mqtt-1.5.0.tar.gz (99kB)  
    100% |#####| 102kB 1.0MB/s
```

(C) Jari Hiltunen

paho-mqtt-kirjasto sisältää mm. seuraavia metodeja:

- connect() – yhdistä MQTT
- disconnect() – sulje yhteys MQTT
- subscribe() – tilaa aihe
- unsubscribe() – lopeta aiheen tilaus
- publish() – lähetä viesti aiheeseen
- .on_message – lisää tapahtuma on_message-tapahtumaan (ns. callback)

Python-koodissa ensin tulee tuoda Paho-MQTT-kirjasto käyttöön.

Se tapahtuu komennolla:

```
import paho.mqtt.client as mqtt
```

Kun kirjasto on tuotu, seuraavaksi luodaan client-objekti. Objektilla pitää olla yksilöllinen ID, kuten objekteilla muutenkin. Objekti luodaan käyttämällä mqtt.Client()-metodia. Metodissa on neljä valinnaista parametriä, mutta yleensä niitä ei tarvita:

```
kanalaLampo = mqtt.Client("kanalalampo_mqtt") # Luodaan MQTT client-objekti
```

Seuraavaksi clientti yhdistetään MQTT-brokeriin (serveriin, tässä raspi itsessään) ja se tapahtuu connect-metodilla. Metodissa on neljä parametria, host (isäntä), port (portti), keep alive (yhteysaika) ja bind address (sidottu osoite), mutta tarvitsemme vain isännän ip-

osoitteen, joka tässä on raspi itsessään eli localhost. Oletusportti on 1883.

```
kanalaLampo.connect("localhost", 1883) # Yhdistetaan MQTT  
brokeriin (serveriin)
```

Julkaisu tapahtuu käyttämällä publish()-metodia, joka sisältää myös muutamia parametreja. Tässä tarvitsee kuitenkin vain aiheen ja viestin:

```
kanalaLampo.publish("Lampotila", 22.3) # Laheta lampotilaviesti  
aiheena Lampotila
```

Aiheen tilaaminen tapahtuu vastaavasti subscribe-metodilla:

```
kanalaLampo.subscribe("Lampotila") # Tilaa aihe Lampotila
```

Tämä vaatii myös erityisen metodin "callback" käyttämisen toimiakseen, sillä callback voisi olla kuten ESP32 mikrokontrollerin keskeytys, joka toteuttaisi MQTT-brokerilta tulevan viestin. Eli ennen looppia pitää olla komento:

```
kanalaLampo.on_message = viestiToiminto # Liitetään  
"viestiToiminta" viestitapahtumaan
```

Luodaan edellä esitelty viestiToiminto-metodi, joka ottaa kolme parametria, clientti, vdata, viesti:

```
def viestiToiminto (clientti, vdata, viesti):
```

Tämän funktion sisällä voimme tehdä mitä haluamme, kuten testata sitä tuleeko jokin tietty arvo ja tehdä asioita sen mukaan. Esimerkiksi voimme printata aiheen mukaiset viestit:

```
def viestiToiminto (clientti, vdata, viesti):  
    aihe = str(viesti.topic)  
    viesti = str(viesti.payload.decode("utf-8"))  
    print(aihe + viesti)
```

Esimerkkikoodi:

```
import paho.mqtt.client as mqtt # Tuo MQTT kirjasto  
import time # aikakirjasto delayta varten
```



```
# Viestitapahtuma
```

```
def viestiToiminto (clientti, vdata, viesti):
```

```
    aihe = str(viesti.topic)
```

```
    viesti = str(viesti.payload.decode("utf-8"))
```

```
    print(aihe + viesti)
```

```
kanalaLampo = mqtt.Client("kanalalampo_mqtt") # Luodaan MQTT  
client objekti
```

```
kanalaLampo.connect("localhost", 1883) # Yhdistetaan MQTT-  
brokeri
```

```
kanalaLampo.subscribe("Lampotila") # Tilataan Lampotila-aiheiset  
viestit
```

```
kanalaLampo.on_message = viestiToiminto # Liitetään edellä esitetty  
viestiToiminto tilaukseen
```

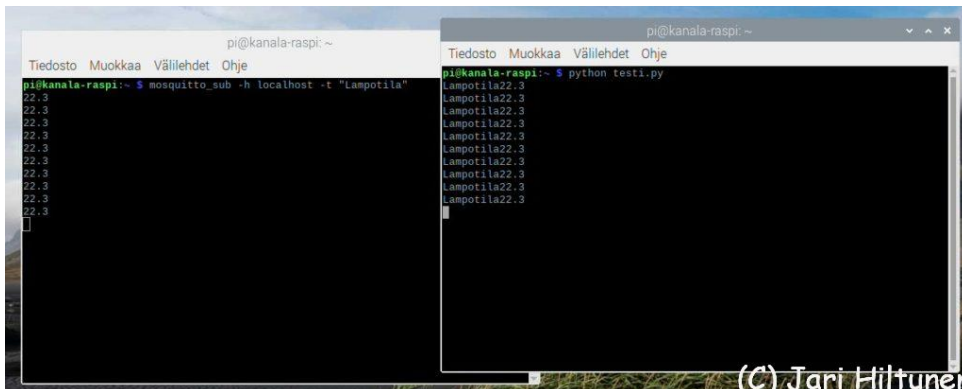
```
kanalaLampo.loop_start() # Käynnistä MQTT clientti
```

```
# Ohjelman looppi
```

```
while(1):
```

```
    kanalaLampo.publish("Lampotila", 22.3) # Julkaise viesti MQTT  
brokerille
```

```
    time.sleep(1) # Venttaa sekunti
```

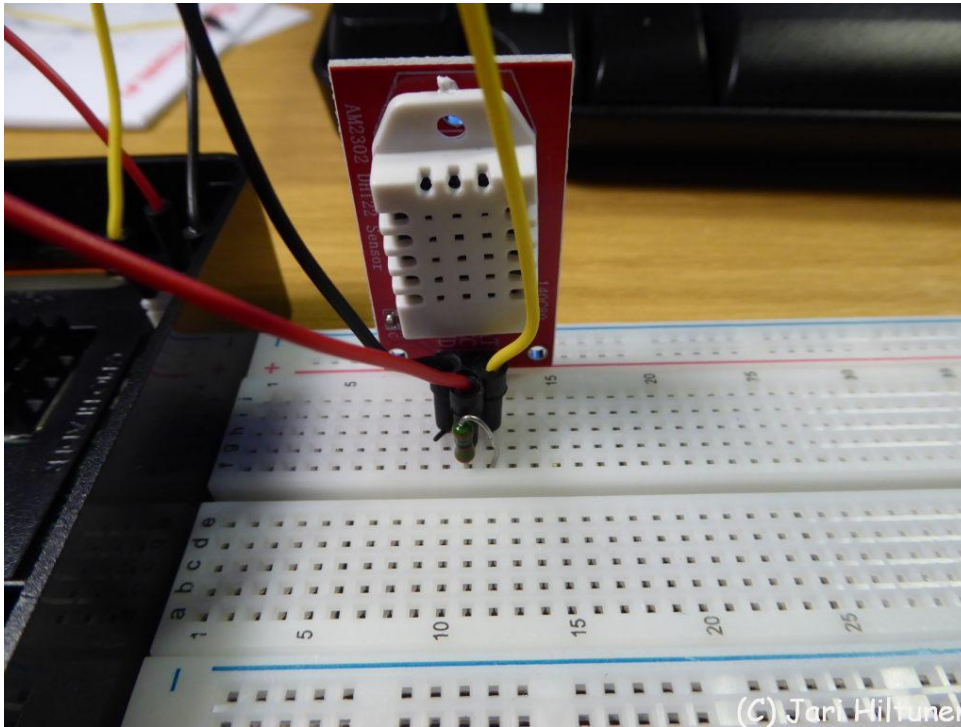


Vasemmalla tilaajan näkymä, oikealla python koodin tuottamat rivit.

DHT-22 kytkentä ja mtqq - testaaminen

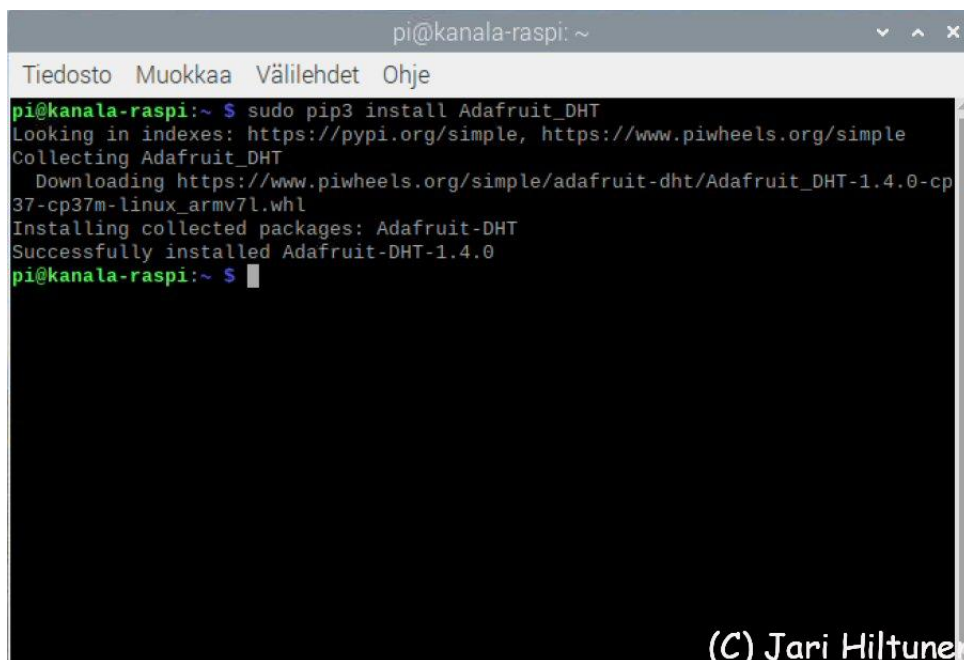
Seuraavaksi testataan miten DHT-22 sensorista saadaan mqtt-viestit Pythonilla brokerille. Tarvitset DHT-22 anturin, ylösvetovastuksen 4,7 kOhm – 10 kOhm väliltä (kuvassa 10 kOhm). Ylösvetovastus kytketään siis datalinjan ja VCC (5V) välille. Kuvassa malli.

Mikäli haluat käyttää raspista lattakaapelia, älä käytä uudempia 40-pinnisiä IDE-kaapeleita, sillä niissä on oikosuljettu joitakin pinnejä yhteen! Raspissa parittomat pinnit menevät piirilevyn sisäpuolella, parilliset piirilevyn ulkolaidalla. Mikäli sinulla ei ole naaras-koiras välijohtoja, voit käyttää koiras-koiras välijohtoa siten että laitat esimerkiksi kiintolevyistä tai jostain muualta irtirevittyjä jumppereita raspin liittimessä ja lyhennät hieman koirasliittimen nastaa siten ettei se yllä piirilevylle saakka kun painat sen jumpperipalikkaan.



DHT-22 anturin kanssa käytetään Adafruit-DHT (Adafruit-Python-DHT)-kirjastoa. Asennus tapahtuu komennoilla:

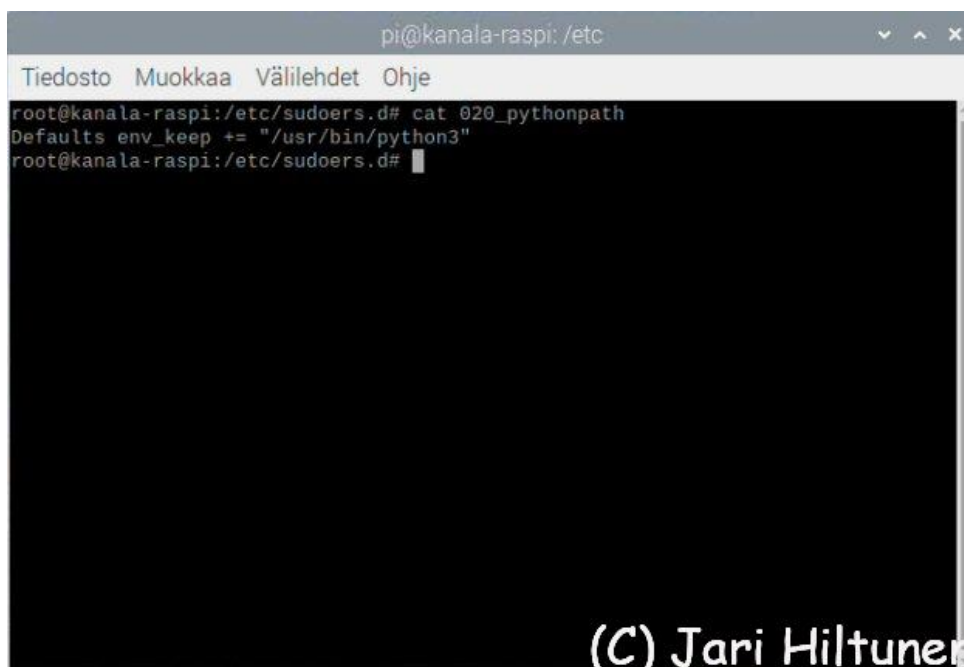
```
sudo pip3 install Adafruit_DHT (tai sudo pip install Adafruit_DHT)
```



```
pi@kanala-raspi: ~  
Tiedosto Muokkaa Välilehdet Ohje  
pi@kanala-raspi:~ $ sudo pip3 install Adafruit_DHT  
Looking in indexes: https://pypi.org/simple, https://www.piwheels.org/simple  
Collecting Adafruit_DHT  
  Downloading https://www.piwheels.org/simple/adafruit-dht/Adafruit_DHT-1.4.0-cp37-cp37m-linux_armv7l.whl  
Installing collected packages: Adafruit-DHT  
Successfully installed Adafruit-DHT-1.4.0  
pi@kanala-raspi:~ $
```

(C) Jari Hiltunen

Tässä kohtaa saatat joutua tekemään sudoers.d hakemistoon tiedoston, esimerkiksi 20_pythonpath, jossa kerrot polun mitä python-versiota käytät. Esimerkiksi:



```
pi@kanala-raspi: /etc  
Tiedosto Muokkaa Välilehdet Ohje  
root@kanala-raspi:/etc/sudoers.d# cat 020_pythonpath  
Defaults env_keep += "/usr/bin/python3"  
root@kanala-raspi:/etc/sudoers.d#
```

(C) Jari Hiltunen

Python-polun saat tietoosi komennolla which python.

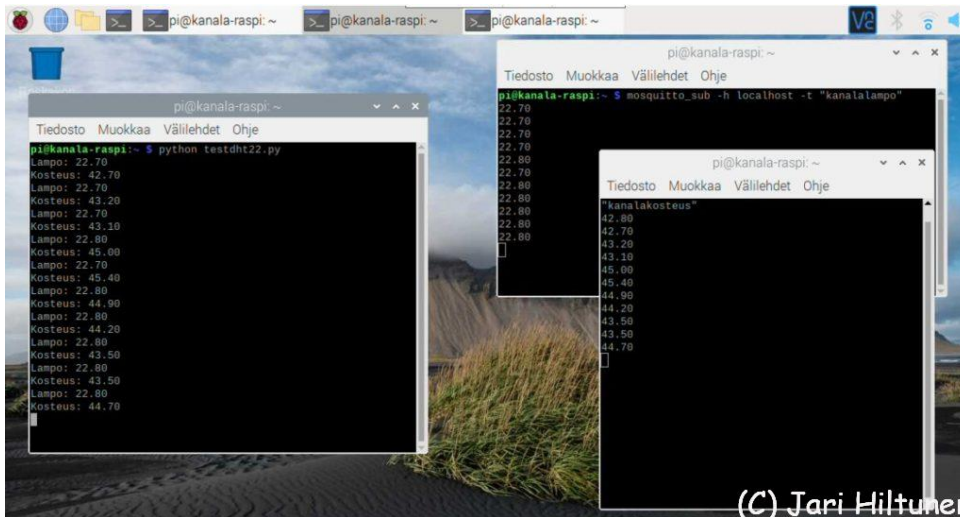
Mikäli poluissa on ongelmaa, valittaa python joko sitä, että sudo python xxxx ei löydä paho-mqtt:tä tai sitten Adafruit_DHT-kirjastoa. Näiden selvittämisessä voi olla viisainta ajaa sekä pip että pip3 install molempiin uudelleen.

Sitten kun kytkennät on tehty ja kirjastot asennettu Pythoniin, voidaan tehdä sovellusta. Alla olevassa kuvassa toimivan

testisovelluksen näyttö, alempana koodi, joka löytyy myös [githubista](#).

Vasemmalla on koodista aktivoituna print, jota ei tietysti normaalissa käytössä tarvita. Tällä voit katsoa sen että anturi todella toimii.

Oikealla on avattuna kaksi eri pääteikkunaa, joista toisessa mtqq-tilaus koskee "kanalalampo"-aihetta ja toinen "kanalakosteus"-aihetta. Kuten huomaat, molempiin tulee samat arvot kuin vasemmalla mtqq-viestejä lähettävässä ruudussa.



```
import paho.mqtt.client as mqtt #mtqq kirjasto
import time
import sys
import Adafruit_DHT #adafruit-kirjasto
```

```
mtqqanturi = mqtt.Client("kanalasisa") #MQTT objekti
mtqqanturi.connect("localhost", port=1883, keepalive
mtqqanturi.loop_start() #Loopin kaynnistys
```

```
kanala_dht22_lampo = "kanalalampo" # subscribe nimi
kanala_dht22_kosteus = "kanalakosteus" #subscribe ni
```

```
while True:
```

```
    try:
```

```
        kosteus22, lampo22 = Adafruit_DHT.read_retry
```

```
        if lampo22 is not None:
```

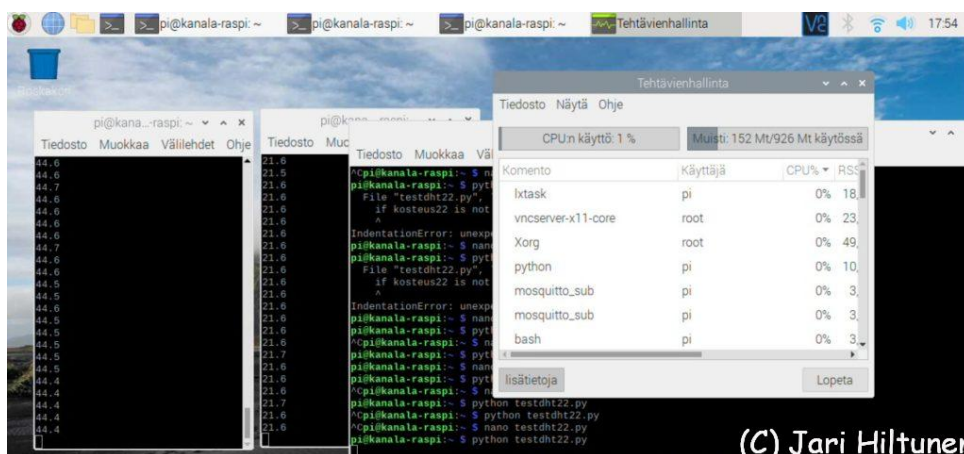
```
            lampo22 = '{:.2f}'.format(lampo22) #des
```

```

        print ("Lampo: ") + str(lampo22)
        mtqqanturi.publish(kanala_dht22_lampo, p
    if kosteus22 is not None:
        kosteus22 = '{:.2f}'.format(kosteus22) #
        print ("Kosteus: ") + str(kosteus22)
        mtqqanturi.publish(kanala_dht22_kosteus,
        time.sleep(3)
    except (EOFError, SystemExit, KeyboardInterrupt)
        mtqqanturi.disconnect()
        sys.exit()

```

Prossessorin suhteen mqtt vaikuttaa hyvältä ratkaisulta. Vaikka brokeri ja clientti pyörii samassa laitteessa, eivät ne käytännössä kuormita raspia juuri ollenkaan. Koska scripti on dedikoitu per laite, ei se oikeastaan kaipaa muuta kuin ehkä crontabilla käynnistyksen, tai sovelluksesta käynnistyksen. Argumenttien välittäminen voisi periaatteessa olla turha riski.



Jatkan scriptin kehittelyä, lisäään csv-tallennusta ja hieman lisää virheiden tarkastusta. Teen erikseen ESP32:lle tarkoitetun bloggauksen heti kun saan kyseisiä härpäkkeitä näppiini ja pääsen koodaamaan.

Koodaukseen sain ideoita mm. Cayenne Communityn [esimerkistä](#).

Seuraavaksi asennan InfluxDB-tietokannan ja Grafanan datan visualisointiin. Tietokantaa tarvitsen muutenkin projektissa. Huomaa, että muutan scriptejä hieman matkan varrella kun kehityn itse näiden asioiden kanssa. Bloggaus asennuksesta [tässä](#).

□ Tagged [adafruit](#), [automaatio](#), [dht22](#), [esp32](#), [iot](#), [kanala](#), [mosquitto](#), [mqtt](#), [nodemcu](#), [pahomqtt](#), [python](#), [raspberrypi](#)

PREVIOUS

[Ryobi RY18PCB-140
puhdistusharja](#)

NEXT

[InfluxDB-tietokannan ja
Grafanan asennus
Raspberrypiyn](#)
