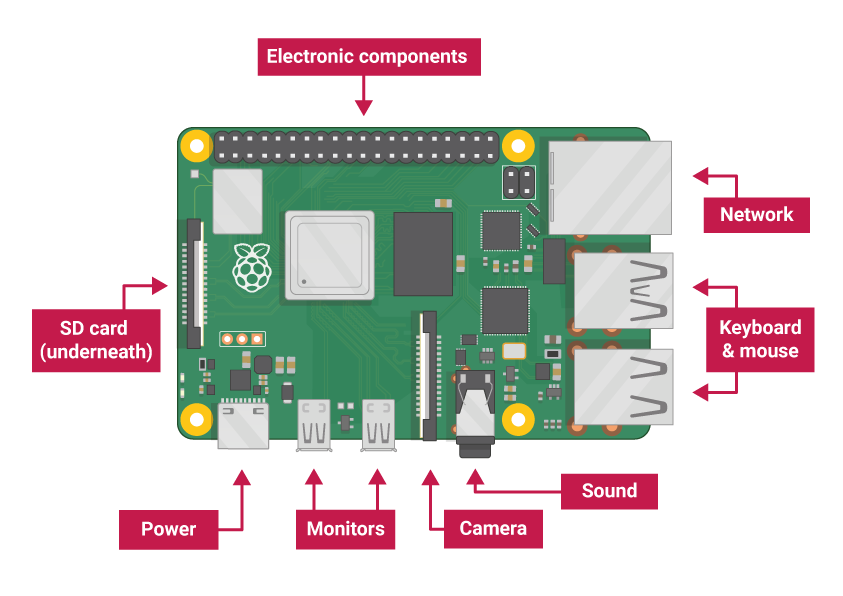
# Mikrokontrollerin anturidatan tallennus pilveen ja luku PC:n MySQL-kantaan

Manne Tervaskanto, OAMK, Syksy 2018

(muokannut Juha Hirvonen, SeAMK, Kevät 2020)



*Raspberry Pin liitännät.*

*Tätä dokumenttia voi käyttää perusopetukseen liittyen anturointiin, mikrokontrollerin käyttöön, API-rajapintaan sekä tietokantoihin lokaalisti sekä pilvessä. Aluksi mikrokontrollerilta luettua lämpötiladataa kirjoitetaan ja visualisoidaan pilvessä. Tarkoituksena on myös, että dataa luetaan pilvestä takaisin esim. PC:n MySQL kantaan. Oletuksena on peruskäsitteet Raspberry Pi:n käytöstä (erityisesti anturien liittäminen GPIO-pinneihin) sekä Linux ja Python ohjelmointikielistä. Lisätietoa Raspberry PI:stä esim. linkistä http://www.toptechboy.com/raspberry-pi-with-linux-lessons/*

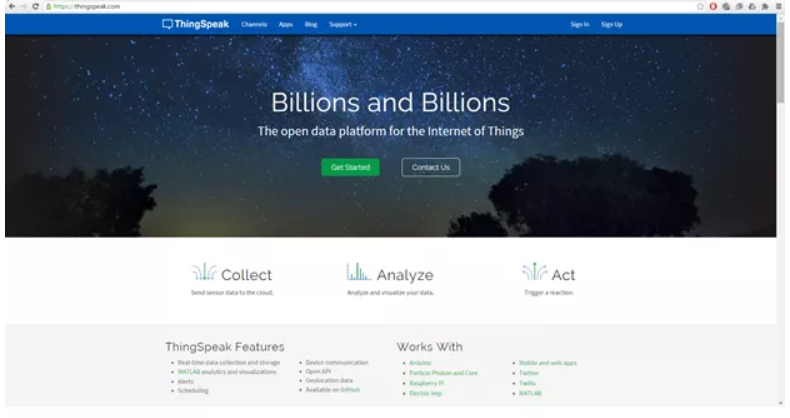
## ThingSpeak pilvipalvelu

Raspberry PI:llä tuotettua dataa olisi hyvä saada tallennettua ja luettua sitten sitä etänä vaikkapa selaimen kautta. Tämä mahdollistaa mm. kodin sensorien (lämpötila, kosteus, yms.) lukemisen etänä mistä tahansa verkkoyhteyden kautta. Tähän tarkoitukseen on olemassa useita kaupallisia pilvipalveluita (Amazon AWS, MS Azure, Google Cloud, yms.). Eräs maksuton palvelu sensoridatan lukemiseen http-protokollalla internetin tai (W)LAN:n kautta on avoimen lähdekoodin alusta nimeltään *Thingspeak.com*.

ThingSpeak:lla on läheinen suhde Mathworks Inc.:iin (Matlab:n myynti, tuki ja tuotekehitys). ThingSpeak-verkkosivuston käyttöönottaminen on mm. mahdollista sallittujen Mathworks-käyttäjätilien avulla. Dokumentaatio sivuston käyttöön ja parametrointiin löytyy myös Mathworksin sivuilta.

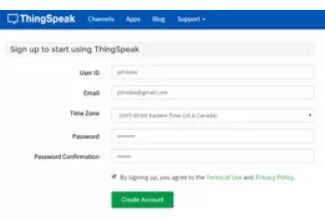
Tässä harjoituksessa otetaan käyttöön käyttäjäkohtainen Thingspeak -sivusto ja kirjoitetaan siihen dataa Raspberry PI:stä. Datana käytetään Raspberryn prosessorin lämpötilaa. Minimipäivityssykli palvelussa datalle yhdessä kentässä on 15 sekuntia.

### Kirjaudu sivustolle ja luo tunnukset



Kuva 1. Thinspeak.com tunnusten luonti

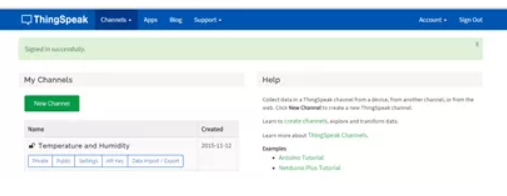
Klikkaa kohtaa ”Sign Up” ja täydennä kohdat.



Kuva 2. Täydennä kohdat.

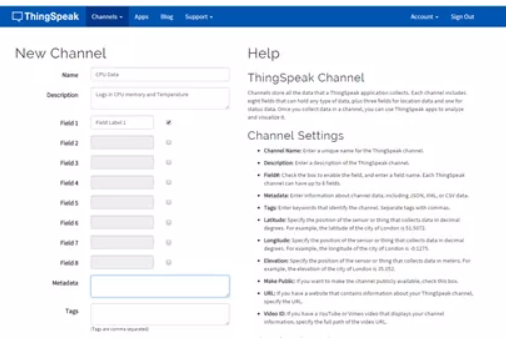
### Kanavan luonti

Kun uusi tili on aktivoitu, luo uusi kanava klikkaamalla ”New Channel”.



Kuva 3. Uusi kanava.

Voit syöttää mittaustiedon (esim. CPU Lämpötila) nimen kenttään 1. Jos haluat lisätä kenttiä, valitse kentän vieressä olevan boksi ja kirjoita dataa kuvaava nimi. Tee kuitenkin vähintään kaksi kenttää.



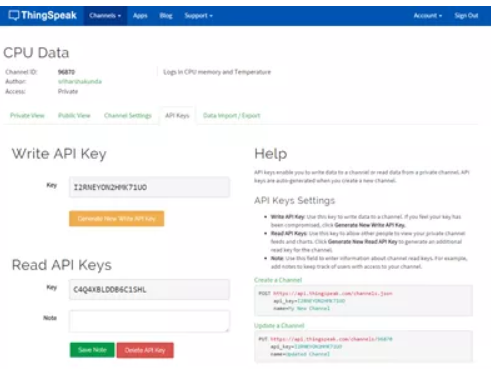
Kuva 4. Kanavan tietojen määritys.

Paina lopuksi ”Save Channel”.

### Määritä API avain (Application Programming Interface eli ohjelmointirajapinta)

Tietojen lähettämiseksi tarvitaan API-avain, johon lisätään myöhemmin Python-koodinpätkä, jonka avulla voidaan lähettää anturitietoja Thingspeak-sivuston tietokantaan.

Napsauta "API Keys" -välilehteä saadaksesi avaimen anturitietojen lähettämiseksi.



Kuva 5. API avaimet, ota talteen **Channel ID** ja **Write API Key**.

### Python koodin modifiointi ja ohjelman suoritus

Esimerkikoodeissa käytetään Pythonin moduuleja *http.client* ja *urllib*. Ensimmäinen sisältää asiakaspuolen http-protokollan mukaiset komennot (GET, POST, PUT, DELETE), ja jälkimmäinen hoitaa URLien kanssa työskennellessä tarvittavat työkalut (URLien lukeminen ja avaaminen yms.)

Muodosta ja tallenna alla oleva CPU\_Temp.py -file oikeaan Raspberryn kansioosi. Lisää **key** kohtaan oma Write API Key. Voit myös muuttaa datan päivityssykliä, mutta et siis alle 15:een sekuntiin.

Tallenna ja aja ohjelma Linuxin puolella. Mikäli ohjelma lähettää dataa palveluun, tulostuu komentoikkunaan lämpötila Celsius-asteina esim. 47.25 ja 200 OK.

Ohjelma lähettää siis Raspin prosessorin lämpötilatietoa pilveen. Tietona voi luonnollisesti olla mikä tahansa muukin anturi, jota Raspi lukee.

from http.client import HTTPConnection

from urllib.parse import urlencode

import time

sleep = 15 # Paivitystaajuus / s

key = 'AVAIN\_TÄHÄN'

# Kerropa Raspin prossun lampotila Celsiuksina

def thermometer():

    while True:

        # Laske Raspin prossun lampotila Celsiuksina

        temp = int(open('/sys/class/thermal/thermal\_zone0/temp').read()) / 1e3 # Lue lampotila

        params = urlencode({'field1': temp, 'key':key })

        headers = {"Content-type": "application/x-www-form-urlencoded","Accept": "text/plain"}

        conn = HTTPConnection("api.thingspeak.com:80")

        try:

            conn.request("POST", "/update", params, headers)

            response = conn.getresponse()

            print(temp)

            print(response.status, response.reason)

            data = response.read()

            conn.close()

        except:

            print("connection failed")

        break

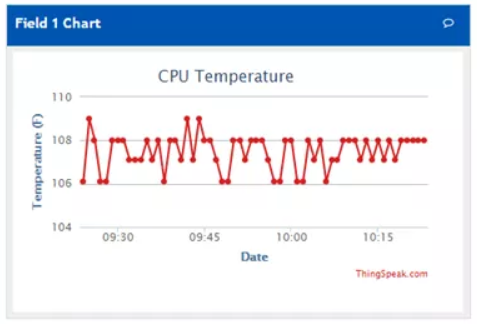
if \_\_name\_\_ == "\_\_main\_\_":

        while True:

                thermometer()

### Tarkista data sivustolta

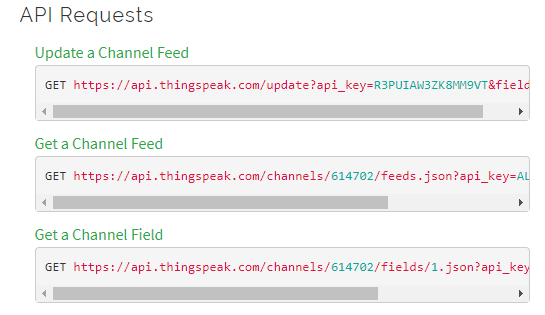
Mene omalle sivustolle (esim. <https://thingspeak.com/channels/614702>) ja tarkista, että dataa kertyy pilven tietokantaan ja se näkyy myös trendinä näytöllä. Voit muuttaa kuvan asetuksia halutessasi. Esimerkiksi kuvassa voidaan näyttää 10 pisteen keskiarvo yhdessä pisteessä ja voit mm. määrittää ajanjakson, kuinka pitkältä ajalta data näytetään. Trendin ja taustan väriä voi muuttaa, jne. Voit myös tallettaa datan esimerkiksi Excelin .csv muotoon.



Kuva 6. Lämpötiladatan trendi omalla sivustollasi.

## Datan luku pilvestä PC:lle

Raspberry PI:llä tuotettua dataa voidaan siis siirtää ThingSpeak pilvialustalle, mutta sitä voidaan myös lukea takaisin omalle koneelle. Mene ensin omalle ThingSpeak sivullesi ja Channels -välilehdellä mene siellä API-keys kohtaan. Sivulla on sekä kirjoitus (Write) ja luku (Read) avaimet. Lisäksi siellä on API Request http-koodit, joilla saadaan luettua dataa pilvestä. Ensimmäinen rivi on kirjoitusta varten ja toinen rivi (Get a Channel Feed) on lukua varten.



Kuva. API Request komennot.

Asenna omalle koneellesi Python 3.7. versio (https://www.python.org/downloads/). Avaa Idle editori ja sieltä New File. Kirjoita siihen seuraavat rivit. Kellattuihin kohtiin laita oma kanavan numero sekä Read API -avain. Mikäli olet tehnyt sivuistasi kaikille näkyvän eli Public View:n, niin tätä api\_key= + lukuavain -kohtaa ei tule koodiin. Voit muuttaa Public ja Private asetuksia Sharing kohdasta.

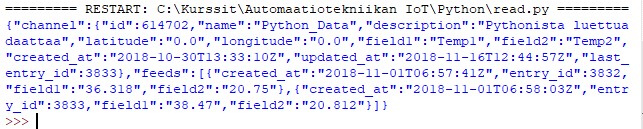
Results 2 rivin lopussa tarkoittaa, että tuloksia näytetään 2 viimeisintä kaikista tauluista (kaikista mittauksista). Voit muuttaa tätä arvoa halutessasi.

from urllib.request import urlopen

websitedata = urlopen("https://api.thingspeak.com/channels/######/feeds.json?api\_key=################&result=2")

print(websitedata.read())

Tallenna koodi aja (Run) F5:sta. Tarkista tulokset Python Shell ikkunasta, kts. alla.



Matlabin sivustolta saat lisätietoa vaihtoehtoisista datan esittämistavoista, kts. <https://se.mathworks.com/help/thingspeak/readdata.html>. Näihin kuuluvat esimerkiksi keskiarvo tietyltä aikajaksolta, jne.

Alla yleinen esimerkki JSON filestä, mistä luetut tietokentät koostuvat (2 mittausta kahdesta taulusta Field1 ja Field2).

The response is a JSON object of the channel feed, for example:

{

"channel": {

"id": 9,

"name": "my\_house",

"description": "Netduino Plus connected to sensors around the house",

"latitude": "40.44",

"longitude": "-79.9965",

"field1": "Light",

"field2": "Outside Temperature",

"created\_at": "2010-12-14T01:20:06Z",

"updated\_at": "2018-01-26T13:08:04Z",

"last\_entry\_id": 13633195

},

"feeds": [

{

"created\_at": "2018-01-26T13:07:48Z",

"entry\_id": 13633194,

"field1": "150",

"field2": "23.014861995753716"

},

{

"created\_at": "2018-01-26T13:08:04Z",

"entry\_id": 13633195,

"field1": "142",

"field2": "23.86411889596603"

}

]

}

Alla vielä Python v2.7. esimerkki (read2.py), mikäli haluat lukea pilvestä 30 sekunnin välein ainoastaan taulun 1 ja 2 viimeisimmän arvon, statustiedon sekä aikaleiman.

from urllib.request import urlopen

import json

import time

READ\_API\_KEY = 'OMA\_AVAIN\_TÄHÄN'

CHANNEL\_ID = 'KANAVAN\_ID\_TÄHÄN'

try:

    while True:

        conn = urlopen("https://api.thingspeak.com/channels/" + CHANNEL\_ID + \

            "/feeds/last.json?api\_key=" + READ\_API\_KEY)

        response = conn.read()

        print("http status code = " + str(conn.getcode()))

        data = json.loads(response)

        print(data['created\_at'])

        print(data['field1'])

        print(data['field2'])

        time.sleep(15)

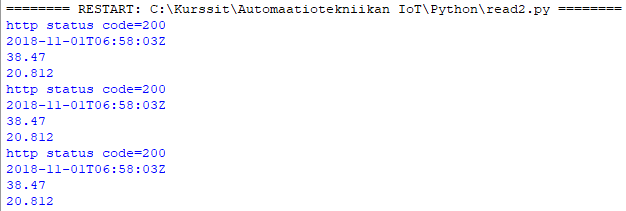
        conn.close()

finally:

    conn.close()

    print("Connection lost")

Tuloksena saadaan yo. koodista seuraavaa (kolme kierrosta). Huomaa, että mittausarvot ja aikaleimat ovat samoja, koska data ei ole muuttunut kannassa ensimmäisen ja viimeisen hakukierroksen välillä.



Mikäli kommunikointi on kunnossa, http status on 200. Virhekoodit saat täältä:

<https://se.mathworks.com/help/thingspeak/error-codes.html>

## PostGres-kannan asennus ja anturitiedon tallennus

1. Lataa ja asenna PostGres 9.6 osoitteesta <https://www.postgresql.org/> (koulun atk-luokissa pitäisi olla valmiiksi asennettuna)
2. Asenna Pythonille moduuli psycopg2 (rajapinta PostGres-tietokantaan) komennolla pip install psycopg2.

Kun PostGres asennus on valmis, tee tietokantaan taulu, johon lisäät aikaleiman ja mittausdatasarakkeet niin monelle mittaukselle kuin on tarvetta:

1. Avaa *pgAdmin* Käynnistä-valikosta (aukeaa selaimeen)
2. Luo uusi tietokanta: Klikkaa oikeaa näppäintä ruudun vasemman reunan valikon kohdassa Databases 🡪 Create
3. Lisää tietokantaan taulu: Klikkaa auki tekemäsi tietokanta 🡪 Schemas 🡪 oikeaa näppäintä kohdassa Tables 🡪 Create
4. Lisää sarakkeita tauluun: klikkaa oikeaa näppäintä tehdylle taululle 🡪 Properties 🡪 Columns 🡪 +
   1. Luo ensin tunnistussarake: anna ensimmäiselle sarakkeelle nimeksi 'id' ja kohtaan Primary key? 🡪 Yes
   2. Luo tarvitsemasi määrä datasarakkeita (katso alla olevaa skriptiä) ja nimeä ne sopivilla nimillä (pienillä kirjaimilla) ja tyypiksi 'character varying'. Pituudeksi talletettavaan dataan nähden sopiva, että muistia säästyy

Alla oleva koodi tallettaa datan ja aikaleiman suoraan PostGres-kantaan, kun ThingSpeak kantaan tulee uusi arvo Raspberryltä. Eli mittausarvo tallennetaan vain kerran MySQL kantaan. Python pollaa ThingSpeak kantaa tässä 5:n sekunnin välein. Mikäli siis uutta dataa on tarjolla, tallennetaan kantaan data, muussa tapauksessa odotetaan 5 sekuntia.

from urllib.request import urlopen

import json

import time

import datetime

from psycopg2 import connect

READ\_API\_KEY = 'OMA\_AVAIN\_TÄHÄN'

CHANNEL\_ID = 'KANAVAN\_ID\_TÄHÄN'

n\_loops = 1  # Silmukkalaskuri, alustetaan 1:een

sleep = 5  # 5 sekunnin odotus

# Luodaan yhteys PostGres-tietokantaan

conn\_to\_db = connect("dbname=tietokantasi\_nimi user=käyttäjätunnuksesi password=salasanasi")

try:

    while True:

        conn = urlopen("https://api.thingspeak.com/channels/" + CHANNEL\_ID + \

            "/feeds/last.json?api\_key=" + READ\_API\_KEY)

        response = conn.read()

        data = json.loads(response)

        time\_from\_db = data['created\_at']

        if n\_loops == 1:

            script\_time\_stamp = time\_from\_db

            print("Started polling new data in ThingSpeak DB from user", CHANNEL\_ID)

            print("Latest timestamp to compare", data["created\_at"])

        else:

            if time\_from\_db != script\_time\_stamp:

                print("Found new data in ThingSpeak DB")

                print(data["created\_at"])

                print(data["field1"])

                print(data["field2"])

                ts = time.time()

                timestamp = datetime.datetime.fromtimestamp(ts).strftime("%Y-%m-%d %H:%M:%S")

                measurement = data["field1"]

                cursor = conn\_to\_db.cursor()

                sql = "INSERT INTO taulun\_nimi (sarake1, sarake2, sarake3) VALUES (%s, %s, %s);"

                val = ("JH", measurement, timestamp)

                cursor.execute(sql, val)

                conn\_to\_db.commit()

                cursor.close()

                conn\_to\_db.close

            if time\_from\_db == script\_time\_stamp:

                print("No new data in ThingSpeak DB")

        running\_time = ((n\_loops\*sleep-sleep) / 60.0)

        print("Script running time", running\_time, "minutes")

        print("Sleeping", sleep, "seconds")

        script\_time\_stamp = time\_from\_db

        n\_loops += 1

        conn.close()

        time.sleep(sleep)

finally:

    conn.close()

    conn\_to\_db.close

    print("Connection lost")

## Lämpötila-anturi

Käytetään harjoituksessa DS18B20 digitaalista lämpötila-anturia, joka kytketään GPIO pinni 4:seen. Tätä vastaa fyysinen pinni numero 7.

Lämpötila-anturi tai anturit käyttävät vain yhtä datakytkentää (OneWire) mikrokontrollerille. Jokaisella anturilla on oma sarjanumeronsa, joten ne eivät häiritse toisiaan vaikka ne kytketään samaan kontrolleriin. Työssä tarvitset hyppylankaa ja kytkentälevyn sekä 4,7 kilo-ohmin vastuksia.

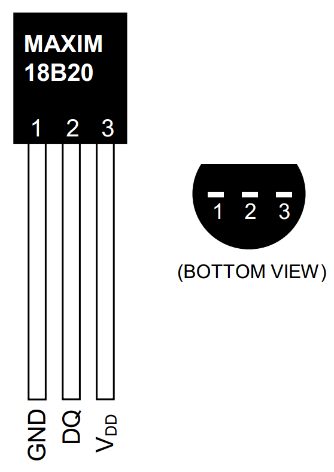
Tarkista aluksi onko **/boot/config.txt** filessä lopussa seuraava rivi, ellei ole lisää rivi sinne ja tallenna tiedosto. Käynnistä uudelleen Raspberry, jotta asetukset tulevat voimaan.

Oletuksena on siis pinni numero seitsemän.

dtoverlay=w1-gpio

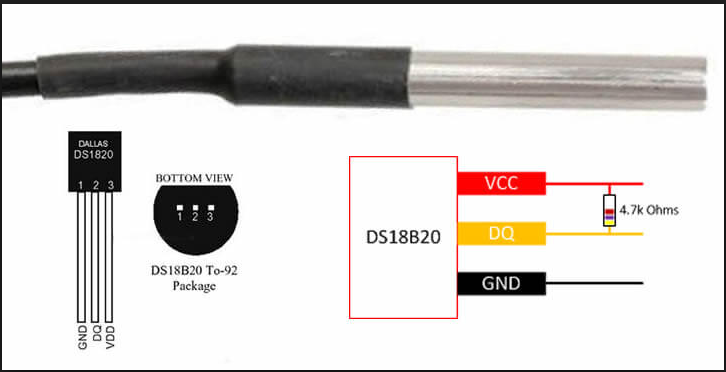
Mikäli käytät muuta pinniä, lisää loppuun ,gpiopin=’pinninumero’. Käytä BCM numerointia edellä olevassa kohdassa.

Tee jälleen uusi tiedosto johon luot uuden tiedoston ja johon teet python ohjelmat. Alla kuva lämpötila-anturista. Huomaa, että kuvaa tulee katsoa metallipinnit osoittamassa katsojaan päin. 3,3:n Voltin plusnapa kytketään GPIO:n ykköspinniin ja toisen reunan pinni miinusnapaan. Data kulkee keskimmäisen pinnin kautta.



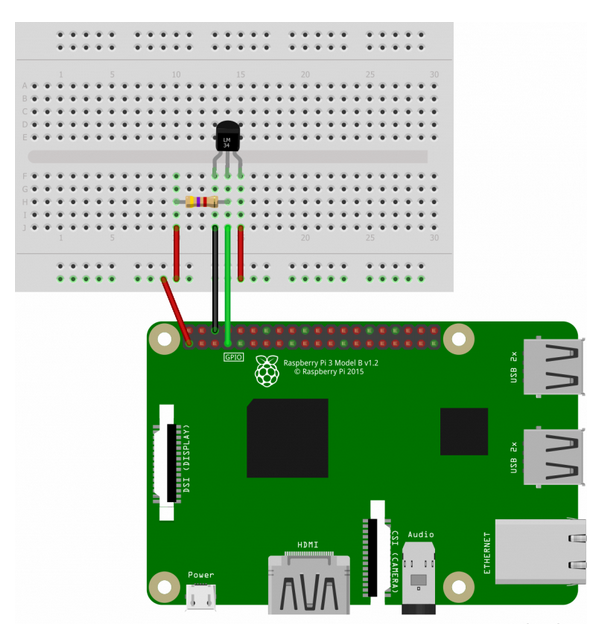
Kuva. DS18B20 lämpötilasensorin kytkennät.

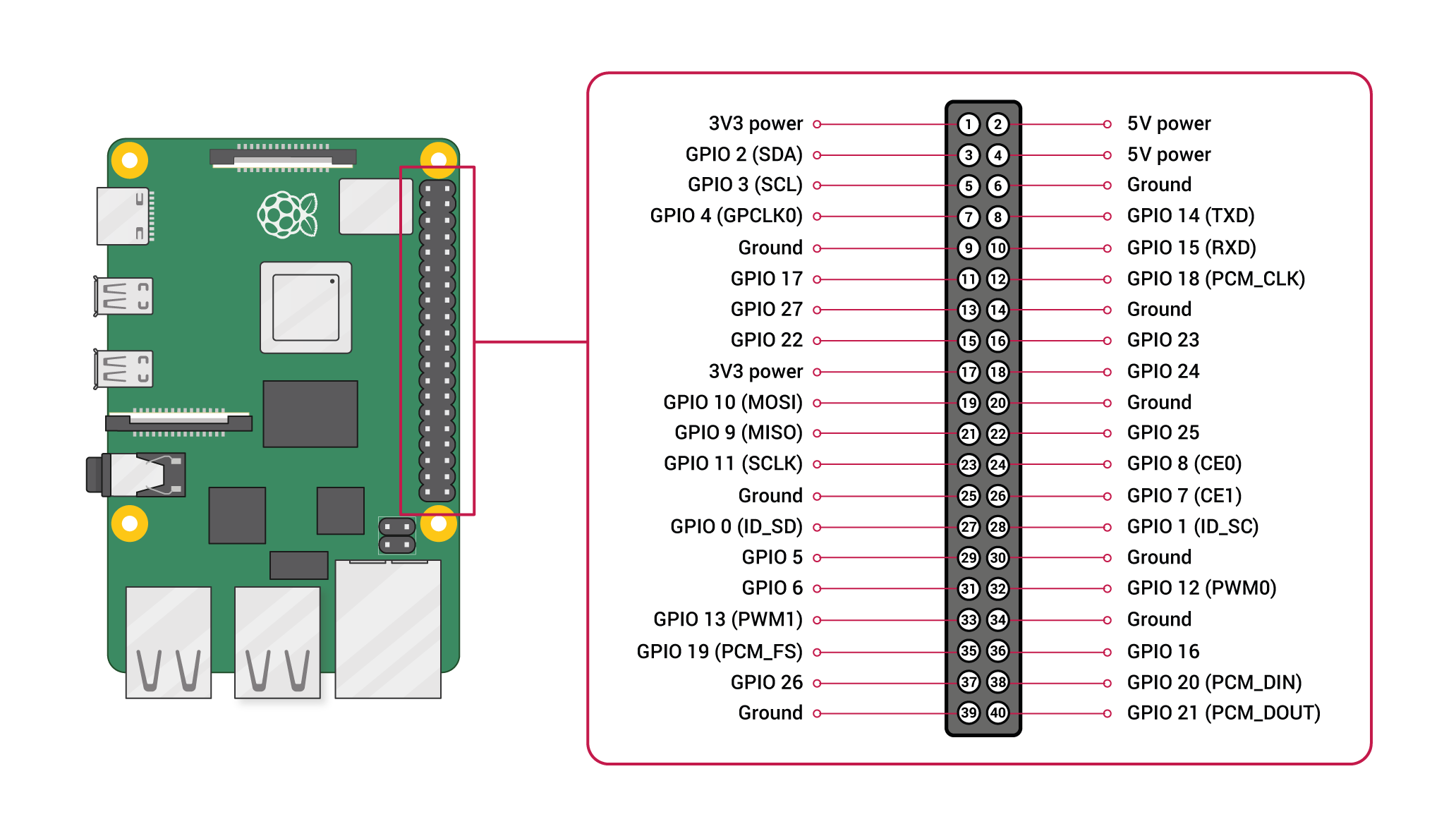
Jos käytössäsi on suojavaipallinen ja kolme johdinta sisältävä DS18B20 anturi, niin plus- (VCC), miinus- (GND) ja datajohtojen (DQ) värit ovat merkittynä alla olevassa kuvassa.

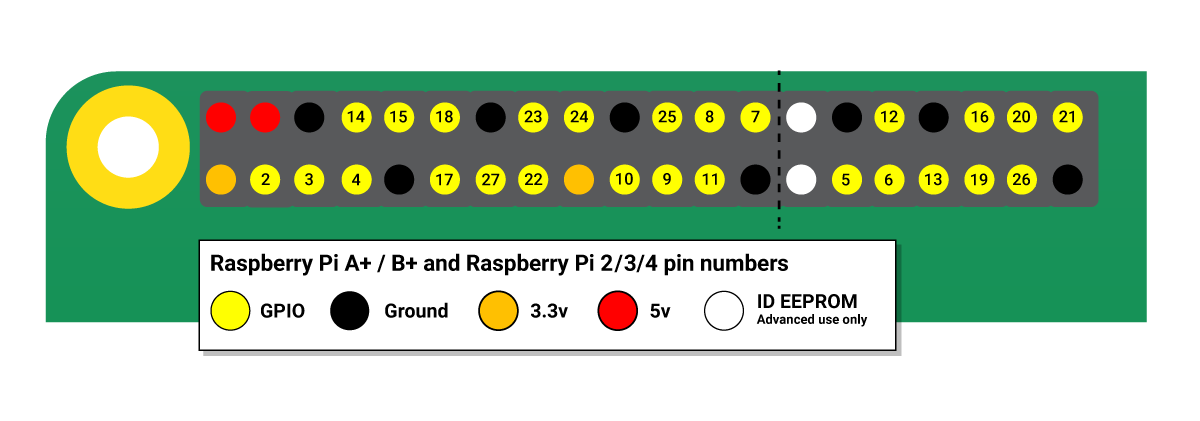


Kuva. Suojavaipallisen DS18B20 lämpötilasensorin kytkennät.

Tee seuraavaksi seuraavanlainen kytkentä.







*Lämpötila-anturin kytkentä kytkentälevyyn.*

Ohjelmoinnissa käytetään aluksi komentoikkunaa aluksi ilman Pythonia. Ladataan ensin Raspberryn käytettäväksi valmiit ajurit. Seuraavat käskyt ajavat 1-wire ja lämpötilasensorin ajurit Raspiin.

sudo modprobe w1-gpio

sudo modprobe w1-therm

Seuraavaksi vaihdetaan laitteen tiedostopolku, jotta nähdään onko mittaus luettavissa oikein.

cd /sys/bus/w1/devices/

ls

Laiteajuri sisältää 28-0xxxxxxxxxx numerosarjan, joka on laitteen uniikki sarjanumero. Mene cd sarjanumero komennolla laiteajurin kansioon, ja tarkastele siellä cat w1\_slave mittausdatan tietoja. Viimeinen luku on lämpötilatieto ilman desimaaleja, esim. 23875 on 23,875 Celsius-astetta. Pidä anturia hetki sormien välissä ja ota uusi luku.

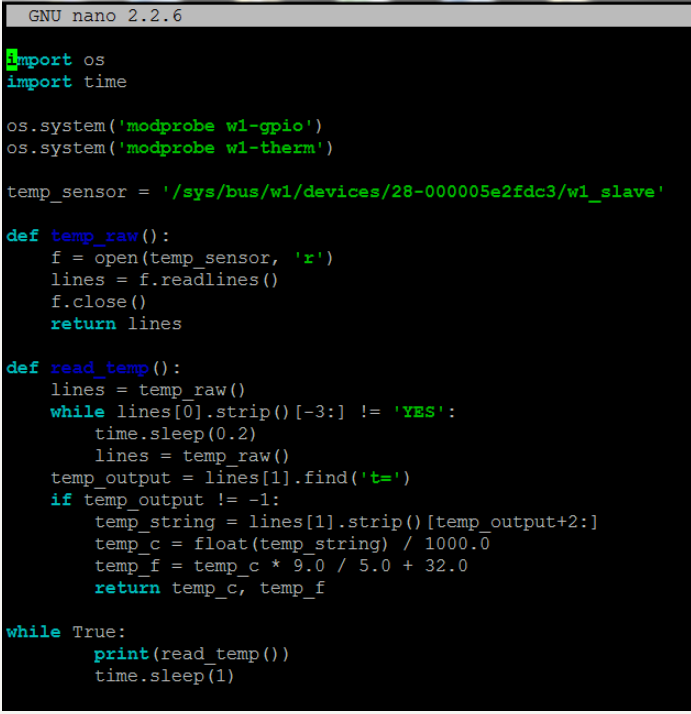
Tee seuraava Python koodi, jossa korvaat sensorin oikealla sarjanumerollasi temp\_sensor kohdassa.

Seuraavaksi tehdään kaksi funktiota def, joita kutsutaan joka kerta while-loopissa.

Ensimmäisen funktion tarkoitus on lukea raakadata, joka sisältää mm. mittauksen. Toinen funktio käyttää ensimmäisen palautusarvoa ja siivoaa siitä raakadatan, jotta jäljelle jää vain mittaustieto. Mittaustieto siivotaan vain jos mittauksen status-tieto on ok, eli YES teksti on saatavissa. Huomaa *eri kuin* merkki on Pythonissa !=. Funktiosta palautetaan sekä Celsius että Fahrenheit tieto.

Lopuksi While-looppi hakee tiedon mittauksesta sekunnin välein funktioiden avulla.

Tallenna koodi ja aja se Linuxissa.



## Sähköpostin lähetys Raspilta

Esimerkissä lähestymisanturin tunnistus aiheuttaa sähköpostin lähettämisen Gmail-osoitteeseen. Muokkaa koodia siten, että sähköpostia lähetetään lämpötila-anturin saavuttaessa tietyn lämpötilan (esim. 30° C; lämmitä anturia sitten kädelläsi, jotta voit testata toiminnallisuutta)

Kts.linkki <https://bit.ly/2IRVG5r>