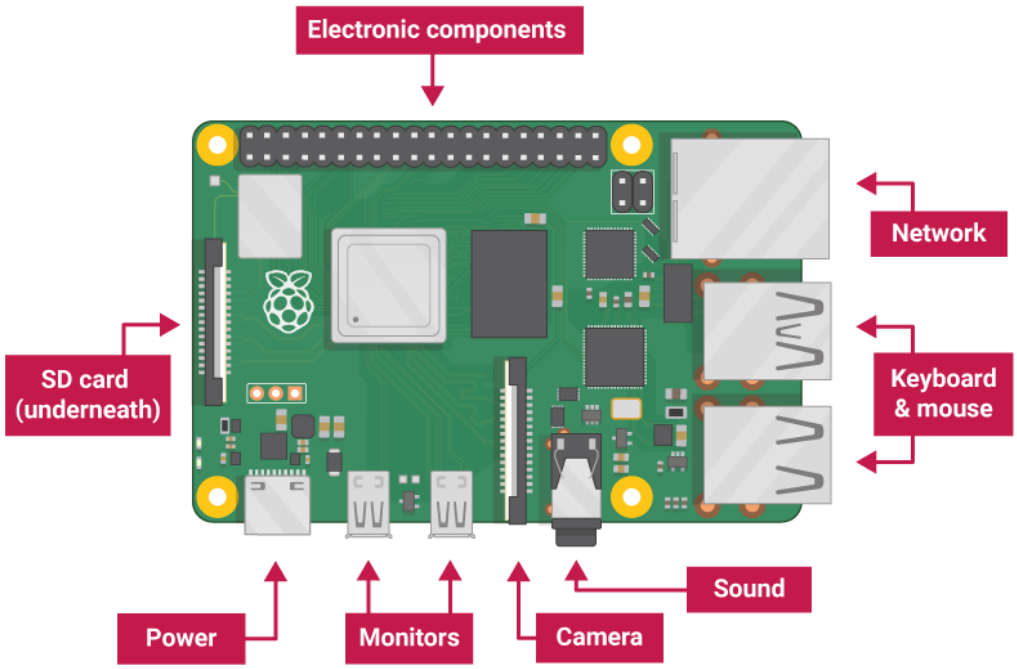
Mikrokontrollerin anturidatan tallennus pilveen ja luku PC:n PostGres-tietokantaan

Juha Hirvonen, SeAMK

Manne Tervaskanto, OAMK

Raspberry Pi

Raspberry Pi (tuttavallisesti Raspi) on halpa yhden piirilevyn tietokone, joka on alun perin tehty opetuskäyttöön. Raspille tehdään usein sovelluksia, joissa se lukee anturidataa ja lähettää sitä eteenpäin pilveen tai toiselle laitteelle, tai huolehtii yksinkertaisesta säädöstä. Kuva alla esittelee Raspin liitännät.



Kuva 1: Raspberry Pin liitännät.

ThingSpeak pilvipalvelu

Raspberry PI:llä tuotettua dataa on kätevä saada tallennettua ja luettua sitten sitä etänä vaikkapa selaimen kautta. Tämä mahdollistaa mm. kodin anturien (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*.

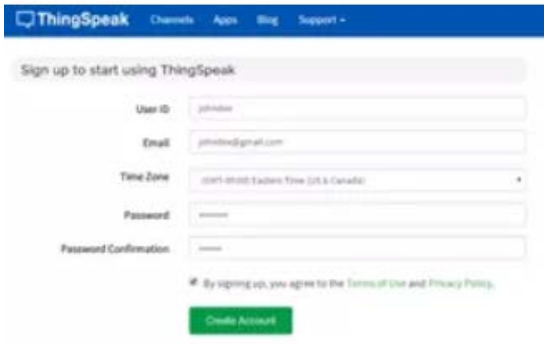
ThingSpeak:lla on läheinen suhde Mathworksiin (Matlabin myynti, tuki ja tuotekehitys). ThingSpeak-verkkosivuston käyttöönottaminen on mm. mahdollista sallittujen Mathworks-käyttäjätilien avulla. Dokumentaatio sivuston käyttöön ja parametrointiin löytyy myös Mathworksin sivuilta.

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 2: ThingSpeak-tunnusten luonti

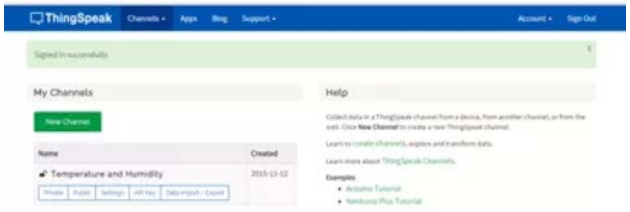
Klikkaa kohtaa ”Sign Up” ja täydennä kohdat.



Kuva 3: Täydennä kohdat.

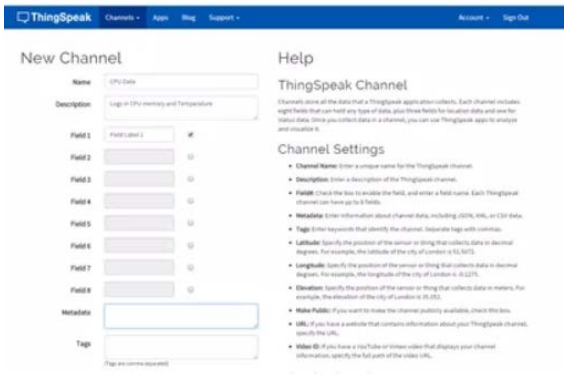
Kanavan luonti

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



Kuva 4: 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.



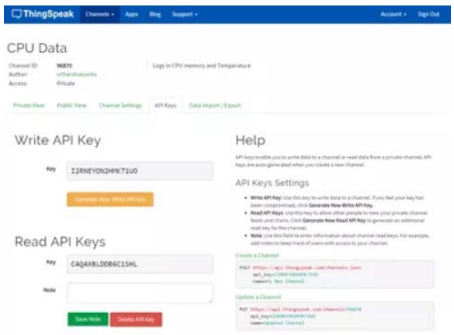
Kuva 5: Kanavan tietojen määritys.

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 6: 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.

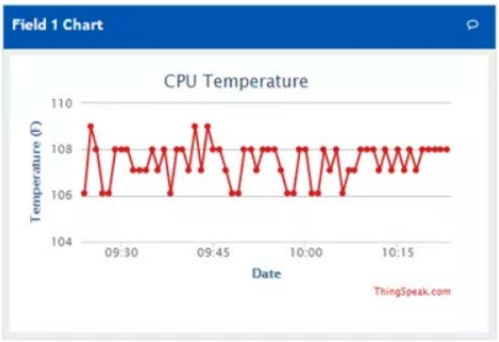
Ohjelma lähettää siis Raspin prosessorin lämpötilatietoa pilveen. Tietona voi luonnollisesti olla mikä tahansa muukin anturi, jota Raspi lukee.

```
from http.client import HTTPConnection
from urllib.parse import urlencode
import time
sleep = 15 # Paivitystaajuus / s
key = 'OMA_AVAIN_TÄHÄN'

# Kerropa Raspin prossun lampotila Celsiuksina
def thermometer():
    # 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")
if __name__ == "__main__":
    while True:
        thermometer()
        time.sleep(sleep)
```

Koodinpätkä 1: Raspin suorittimen lämpötilan lukeminen ja lähettäminen ThingSpeakiin

Tarkista data sivustolta

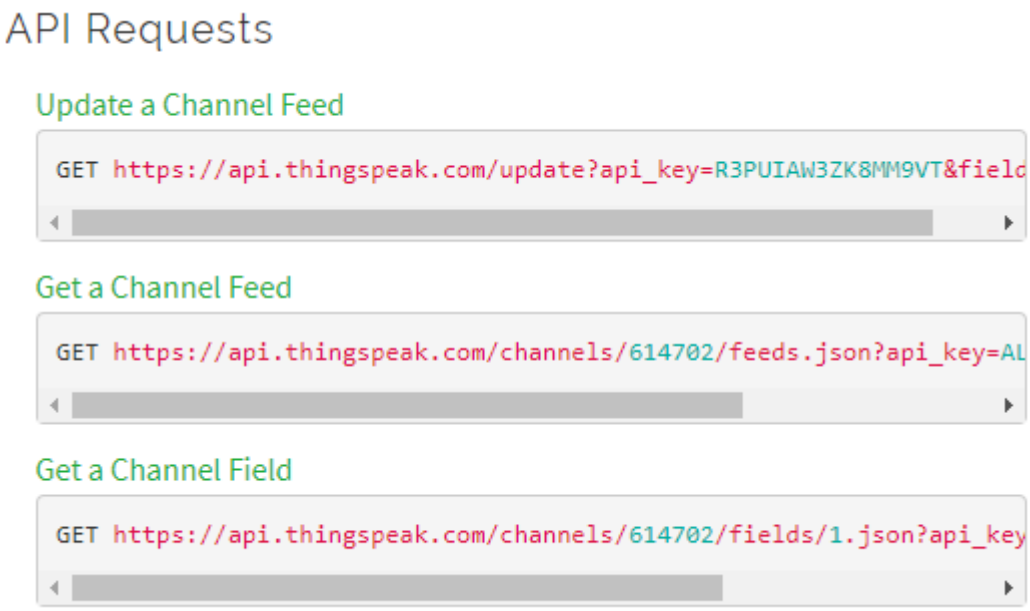
Mene omalle sivustolle (esim. https://thingspeak.com/channels/614702) ja tarkista, että dataa kertyy pilven tietokantaan, ja se näkyy myös trendinä näytöllä. Voit muuttaa kuvan asetuksia halutessasi. Esimerkiksi kuvassa voidaan näyttää 10 pisteen keskiarvo yhdessä pisteessä ja voit mm. määrittää ajanjakson,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



Kuva 7: Lämpötiladatan trendi omalla sivustollasi.

Datan luku pilvestä PC:lle

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



Kuva 8: API request -komennot

Avaa Visual Studio Code ja tee uusi Python-tiedosto. 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())
```

Koodinpätkä 2: Datan luku pilvestä

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

```
========= RESTART: C:\Kurssit\Automaatiotekniikan IoT\Python\read.py =========
{"channel":{"id":614702,"name":"Python_Data","description":"Pythonista luettua
daattaa","latitude":"0.0","longitude":"0.0","field1":"Temp1","field2":"Temp2",
"created_at":"2018-10-30T13:33:10Z","updated_at":"2018-11-16T12:44:57Z","last_
entry_id":3833},"feeds":[{"created_at":"2018-11-01T06:57:41Z","entry_id":3832,
"field1":"36.318","field2":"20.75"},{"created_at":"2018-11-01T06:58:03Z","entr
y_id":3833,"field1":"38.47","field2":"20.812"}]}
>>>
```

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-esimerkki, 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")
```

Koodinpätkä 3: Viimeisten arvojen lukeminen pilvestä

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.

```
======== RESTART: C:\Kurssit\Automaatiotekniikan IoT\Python\read2.py ========
http status code=200
2018-11-01T06:58:03Z
38.47
20.812
http status code=200
2018-11-01T06:58:03Z
38.47
20.812
http status code=200
2018-11-01T06:58:03Z
38.47
20.812
```

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. Luo virtuaaliympäristö Teamsista löytyvän ohjeen *Ohje_virtuaaliympäristö.pdf:*n mukaan ja asenna Pythonille moduuli psycopg2 (rajapinta PostGres-tietokantaan)

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 → +
    a. Luo ensin tunnistussarake: anna ensimmäiselle sarakkeelle nimeksi 'id', tyypiksi ’integer’ ja kohtaan Primary key? → Yes
    b. 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']
```

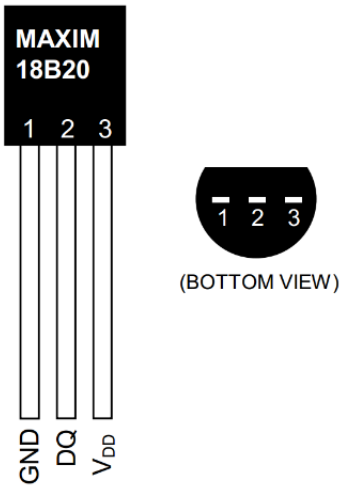
```
        if n_loops == 1:
            script_time_stamp = time_from_db
            print("Started polling new data in ThingSpeak DB from user", CHANNEL_ID)
            print("Latest timestamp to compare", data["created_at"])
        else:
            if time_from_db != script_time_stamp:
                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")
```

Koodinpätkä 4: Datan kirjoitus PostGres-tietokantaan

Lämpötila-anturi

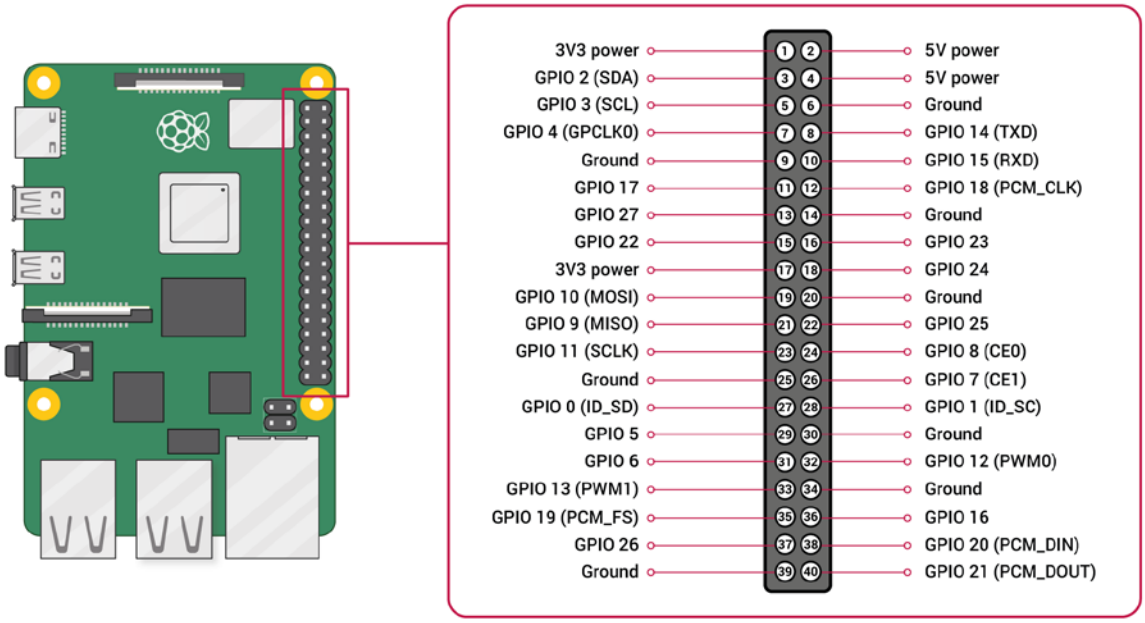
Nyt olemme kokeilleet datan lähettämistä pilvipalveluun sekä lukemista sieltä Raspin integroidun lämpötila-anturin avulla. Kytketään seuraavaksi Raspiin ulkoinen anturi. Käytetään harjoituksessa digitaalista lämpötila-anturia DS18B20. Kyseessä on halpa perusanturi, joka on laajasti saatavilla.

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 koekytkentälevyn, hyppylankaa sekä 4,7 kilo-ohmin vastuksen. Kuva 9 esittelee lämpötila-anturin kytkennät



Kuva 9: DS18B20-lämpötila-anturin kytkennät.

Kuva 10 taas esittelee Raspin GPIO:n kytkennät. Huomaa, että Raspista saa joko 5 tai 3,3 voltin käyttöjännitteen ulkoiselle elektroniikalle (kumpaakin on kaksi kappaletta). Sen lisäksi GPIO:ssa on kahdeksan maapinniä ja 26 pinniä, joita voi käyttää sekä lähtöinä että tuloina.

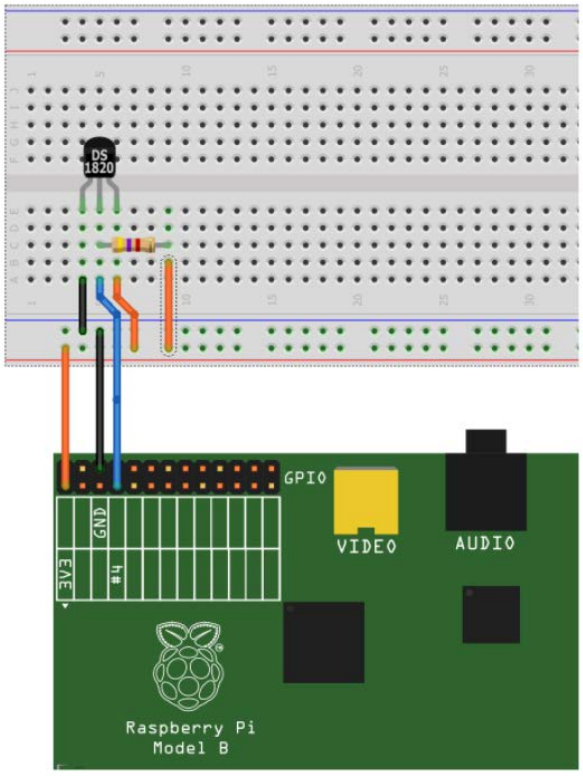


Kuva 10: Raspin GPIO:n kytkennät.

Lämpötila-anturi kytketään kytkentälevyn kautta Raspiin kuten Kuva 11 näyttää. Anturin vasen pinni siis kytketään Raspin maahan, oikealle annetaan 3,3 voltin käyttöjännite, ja keskimmäinen eli datapinni yhdistetään Raspin GPIO:n pinni numero 4:ään 4,7 kΩ:n ylösvetovastuksen kautta. GPIO:n pinni numero 4 on siis fyysinen pinni numero 7 (fyysinen numerointi on kirjoitettu yllä olevan kuvan oikeaan puoliskoon pinnien päälle).

Sammuta Raspista virta ja irrota virtalähde kytkennän ajaksi. Kuvasta 11 poiketen Seamkin sulautettujen järjestelmien laboratoriossa käytettävien koekytkentälevyjen reikärivit – ei sarakkeet – ovat sähköisesti yhteydessä toisiinsa. Ala- ja ylärivin jännitteelle ja maalle tarkoitettujen rivien

reiät on yhdistetty niin pitkälle kuin sininen tai punainen viiva on vedetty. Näin yksi kytkentälevyyn kytketty jännitelähde saadaan helposti kytkettyä useampaan kytkentälevyyn kytkettyyn komponenttiin.



Kuva 11: Lämpötila-anturin kytkentä kytkentälevyyn.

Kun olet saanut kytkennän tehtyä, käynnistä Raspi ja avaa komentokehote. Tarkista aluksi onko konfiguraatiotiedoston **/boot/config.txt** lopussa seuraava rivi

```
dtoverlay=w1-gpio
```

Tarkastele siis tiedostoa komennolla

```
cat /boot/config.txt
```

Jos lopussa ei ole edellä mainittua riviä, avaa tiedosto tekstieditoriin pääkäyttäjän (sudo = superuser do) oikeuksilla komennolla

```
sudo nano /boot/config.txt
```

ja tee tarkistus. Ellei ole, lisää rivi sinne ja tallenna tiedosto. Käynnistä Raspi uudelleen, jotta asetukset tulevat voimaan. Oletuksena on siis (fyysinen) pinni numero seitsemän. Mikäli jostain syystä käytät muuta pinniä, lisää loppuun `,gpiopin='pinninumero'`. Tässä pinnin numero on GPIO:n – ei fyysisen numeroinnin – mukainen.

Ladataan seuraavaksi Raspberryn käytettäväksi valmiit ajurit. Seuraavat käskyt ajavat Raspille ajurit OneWirelle sekä lämpötila-anturille.

```
sudo modprobe w1-gpio
sudo modprobe w1-therm
```

Siirrytään Raspin laitehakemistoon, tulostetaan tiedostolistaus ja tarkistetaan, 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 (saat automaattitäydennyksen painamalla tabulaattoria: kirjoita siis `cd 28` ja paina tabulaattoria, niin et joudu kirjoittamaan koko sarjanumeroa), ja tarkastele siellä mittausdatan tietoja komennolla `cat w1_slave`. 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 anturin oikealla sarjanumerollasi `temp_sensor` kohdassa.

Koodissa on kaksi funktiota (`temp_raw` ja `read_temp`), joita kutsutaan päättymättömässä while-silmukassa sekunnin välein. Ensimmäisen funktion (`temp_raw`) tarkoitus on lukea raakadata, joka sisältää mm. mittauksen. Toinen funktio (read_temp) käyttää ensimmäisen palautusarvoa ja siivoaa siitä ylimääräisen datan, jolloin jäljelle jää vain mittaustieto. Mittaustieto siivotaan vain jos mittauksen status-tieto on ok, eli YES-teksti on saatavissa. Huomaa *eri kuin* -merkki on Pythonissa on `!=`.

Kirjoita koodi, tallenna se ja aja se.

```
import os
import time
os.system('modprobe w1-gpio')
os.system('modprobe w1-therm')

temp_sensor = '/sys/bus/w1/devices/28-0xxxxxxxxxx/w1_slave'

def temp_raw():
    f = open(temp_sensor, 'r')
    lines = f.readlines()
    f.close()
    return lines

def read_temp():
    lines = temp_raw()
    while lines[0].strip()[-3:] != 'YES':
        time.sleep(0.2)
        lines = temp.raw()
    temp_output = lines[1].find('t=')
    if temp_output != -1:
        temp_string = lines[1].strip()[temp_output+2:]
        temp_c = float(temp_string) / 1000.0
```

```
        return temp_c

while True:
    print(read_temp())
    time.sleep(1)
```

Koodinpätkä 5: Lämpötilan lukeminen GPIO:hon kytketyltä lämpötila-anturilta

Anturilta luetun lämpötilan lähettäminen pilveen

Tee uusi ohjelma, jossa muokkaat Koodinpätkiä 1 ja 5 siten, että saat lähetettyä ThingSpeak-pilveen sekä prosessorin että lämpötila-anturin lämpötilat. ThingSpeakin saman kanavan useammalle kentälle saat lähetettyä arvoja näin:

```
params = urlencode({'fieldX': eka_muuttuja, 'fieldY':toka_muuttuja,
'fieldZ':kolmas_muuttuja, 'key':key })
```

Varoitus puhelimeen ThingSpeakista IFTTT:tä käyttämällä

ThingSpeakiin voi lisätä lukuisia lisäosia, joilla siihen saadaan reaktiivisuutta. Jos nyt jättäisit Raspin ja lämpötila-anturin mittaamaan vaikkapa jonkin huoneen tai laitteen lämpötilaa, sinun pitäisi käydä aika ajoin tarkastamassa ThingSpeakin graafeista, että lämpötila on normaalilla alueella. Paljon kätevämpäähän olisi, jos saisit automaattisesti ilmoituksen siitä, että lämpötila ei ole raja-arvojen sisällä.

Hyödynnä oheista esimerkkiä ja IFTTT-palvelua (If This Then That) siten, että saat kännykkääsi ilmoituksen, kun lämpötila-anturin mittaus ylittää arvon 25° C. Lämmitä anturia sitten sormiesi välissä, jotta voit testata toiminnallisuutta

Esimerkki: https://www.mathworks.com/help/thingspeak/use-ifttt-to-send-text-message-notification.html

Sähköpostin lähetys Raspilta

Koska Rapsi on yhteydessä internetiin, sen voi säätää lähettämään varoituksen myös suoraa ilman koko pilvipalvelua. Seuraavassa esimerkissä lähestymisanturin tunnistus aiheuttaa sähköpostin lähettämisen Gmail-osoitteesta. Hyödynnä oheista esimerkkiä ja rakenna ohjelma, joka lähettää sähköpostiin varoituksen, kun lämpötila-anturin mittaama lämpötila ylittää arvon 25° C.

Esimerkki: https://bit.ly/2IRVG5r

Yllä olevassa esimerkissä sähköpostin aihekenttä jää tyhjäksi. Alla olevaa esimerkkiä soveltamalla saat viestillesi lisättyä aiheen:

https://www.techbeatly.com/2019/06/sending-email-using-python-and-smtplib-quick-howto.html/#.XkaDfkFS9EY

Datan luku pilvestä ja sähköpostin lähetys

Lisää Koodinpätkä 3:een varoitussähköpostin lähetys Gmail-tiliä käyttämällä kuten edellä. Raspi siis lähettää lämpötilaa pilveen ja viimeisin arvo luetaan tietokoneella Pythonin-avulla. Jos pilvestä luettu lämpötila-arvo on korkeampi kuin 25° C, ohjelma lähettää varoitussähköpostin.