Lämpötilan mittaus DS18B20 anturilla ja Raspberry PI:llä

Tässä projektissa mitataan lämpötilaa DS18B20 sensorin avulla. Käytössä on vedenkestävä versio sensorista, koska siten sillä on enemmän käyttöä esimerkiksi kotiolosuhteissa.

Python-ohjelma lukee lämpötilaa 5 minuutin välein ja tallettaa arvot MySql-tietokantaan. Lämpötiloista tulostetaan kuvaaja html-sivulle. Tarkoituksena on vielä kehittää projektia siten, että lämpötilat luetaan NodeJs:llä ja talletetaan MongoDB-tietokantaan. Sen jälkeen ne voisi tulostaa html-sivuun expressin avulla.

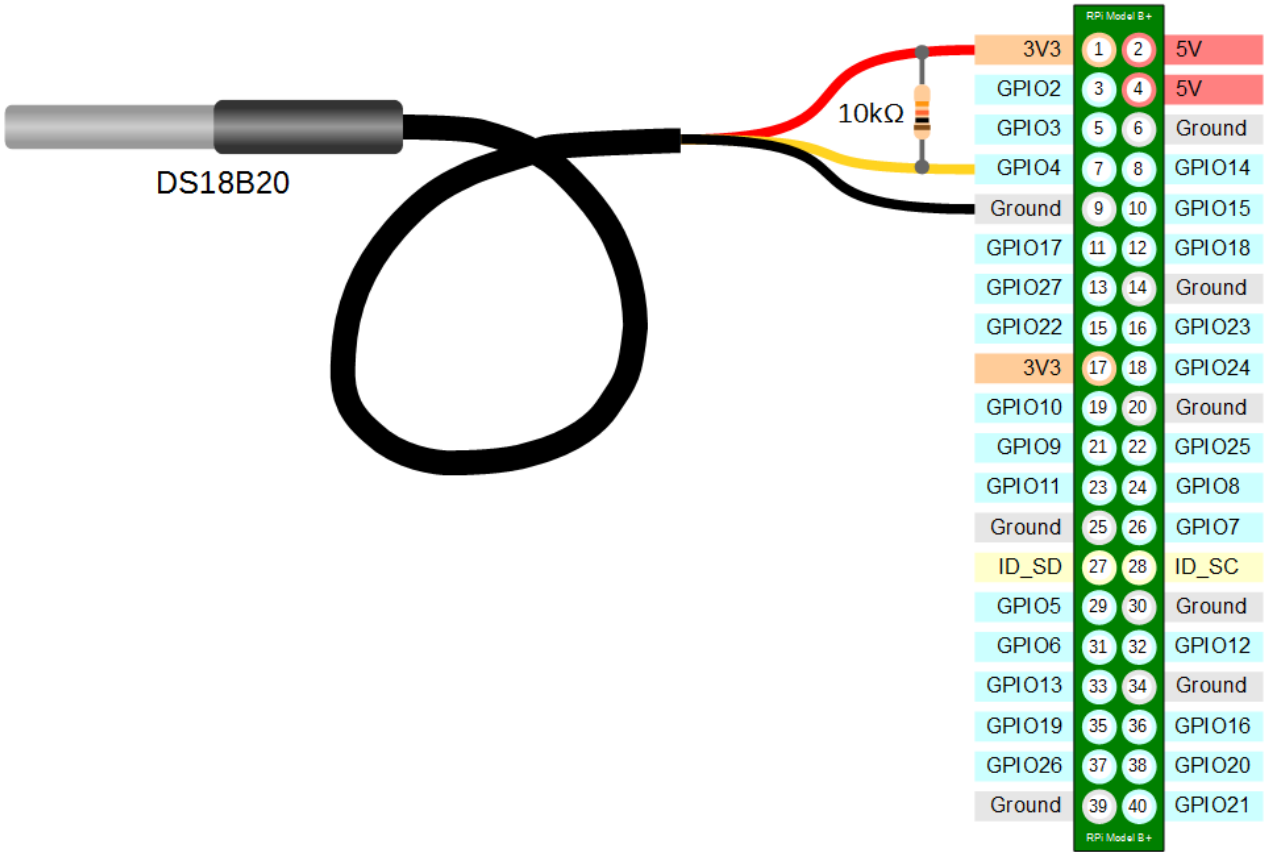
Tarvittavat osat ja komponentit:

- Raspberry PI, USB näppäimistö ja USB hiiri
- DS18B20 sensori (vedenkestävä)
- 10 kOhmin tai 4.7 kOhmin vastus
- Adafruit Raspberry PI GPIO T-Cobbler breakout kit
- koekytkentäalusta
- hyppylankoja

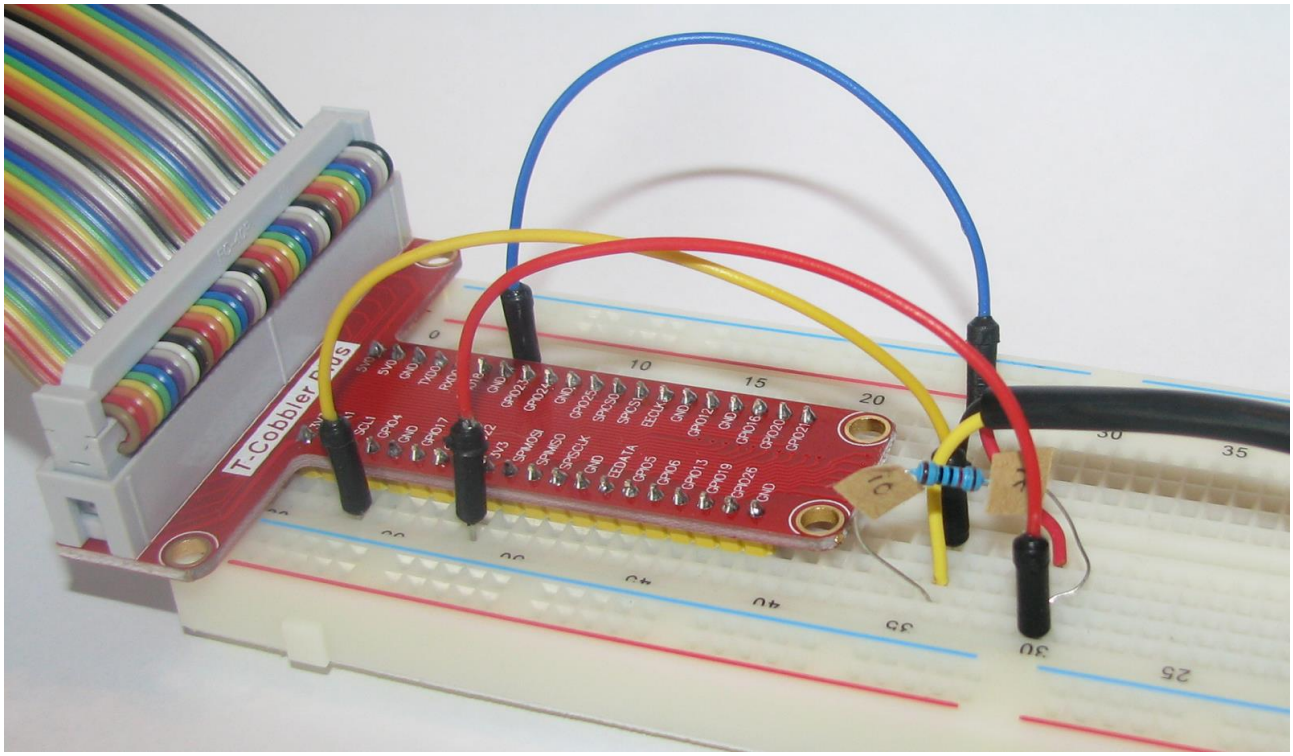
DS18B20 sensorin kytkentä

DS18B20 sensorin musta johdin kytketään Ground pinniin, punainen johdin 3V3 pinniin ja keltainen johdin GPIO4 pinniin kuvan 1 mukaisesti. 4.7 kOhmin tai 10 kOhmin vastus kytketään 3V3 ja GPIO4 pinnien väliin. Se toimii ns. pull-up vastuksena. Täytyy käyttää nimenomaan GPIO4 pinniä, koska Raspberry PI:n käyttöjärjestelmä olettaa, että sitä käytetään 1-wire-kytkennässä.

Jos käytössä on Raspberry PI GPIO T-Cobbler, niin kytkentä näyttää kuvan 2 mukaiselta. Kuvassa 2 lattakaapeli liitetään Raspberryn liittimeen. T-Cobbler helpottaa pinnien kytkemistä oikein.



Kuva 1: DS18B20 1-wire kytkentä Raspberry PI:n pinneihin



Kuva 2: DS18B20 sensorin kytkentä koekytkentäalustaan ja T-Cobbleriin

Raspberry PI ja DS18B20 sensorin lämpötilan lukuun tarvittavat asetukset

Ennen lämpötilan lukemista Python-koodilla tulee tehdä muutama asetus Raspberry PI:lle.

1. Kirjoita kotihakemiston komentokehotteessa sudo nano /boot/config.txt.
2. Kirjoita tiedoston loppuun yksi rivi dtoverlay=w1-gpio
3. Talleta tiedosto ja käynnistä Raspberry PI uudelleen komennolla sudo reboot.
4. Kun Raspberry on käynnistynyt uudestaan, niin aja modprobe komento sudo modprobe w1-gpio kotihakemistossa. Tämä komento rekisteröi uuden sensorin, jotta Raspberry PI tietää, että jokin 1-wire laite on kytketty GPIO liitäntään.
5. Aja sitten komento sudo modprobe w1-therm.Sen jälkeen Raspberry PI tietää mitata lämpötilaa 1-Wire kytkennällä.
6. Siirry devices hakemistoon cd /sys/bus/w1/devices ja aja komento ls.
7. ls komento näyttää listan /sys/bus/w1/devices hakemiston sisällöstä. Siellä pitäisi näkyä hakemisto, joka alkaa 28-xxxxxxxxxx. X:n paikalla on sensorin sarjanumero.
8. Siirry hakemistoon komennolla cd 28-xxxxxxxx. Kirjoita xxxxx :n paikalle sensorin sarjanumero.
9. Tulosta näytölle w1_slave tiedoston sisältö komennolla cat w1_slave. Tulostuksen tulisi näyttää seu raavan kaltaiselta:

```
73 01 4b 46 7f ff 0d 10 41 : crc=41 YES
73 01 4b 46 7f ff 0d 10 41 t=23187
```

10. Jos näkyy YES, se tarkoittaa onnistunutta CRC tarkistusta. Toisellä rivillä näkyy lämpötila t=lukema. Jaa lukema tuhannella, niin saat lukeman celcius-asteina. Esimerkissä 23187 tarkoittaa 23.187 C-astetta.

Lämpötilatietojen lukeminen sensorilta ja tallennus tietokantaan

Seuraava Python-skripti lukee lämpötilatietoja ja tallettaa ne MySQL-tietokantaan. Talleta tiedosto kotihakemistoon ja aja se komennolla

```
sudo python temperature_measurement_to_db.py
```

```
#!/usr/bin/python
# -*- coding: utf-8 -*-

import os
import glob
import time
import MySQLdb as mdb

os.system('modprobe w1-gpio')
os.system('modprobe w1-therm')

base_dir = '/sys/bus/w1/devices/'
device_folder = glob.glob(base_dir + '28*')[0]
device_file = device_folder + '/w1_slave'

def read_temp_raw():
    f = open(device_file, 'r')
    lines = f.readlines()
```

```
    f.close()
    return lines

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

while True:

    try:
        pi_temp = read_temp()
        con = mdb.connect('localhost', 'pi_insert', 'xxxxxxxxxx', 'measurements'
);
        cur = con.cursor()
        cur.execute("""INSERT INTO temperature(temperature) VALUES(%s)""", (pi_t
emp))
        con.commit()

    except mdb.Error, e:
        con.rollback()
        print "Error %d: %s" % (e.args[0],e.args[1])
        sys.exit(1)

    finally:
        if con:
            con.close()

 time.sleep(10)
```

MySQL tietokanta, lämpötilatietojen tallennus ja tulostus html-sivulle

HUOMIO! Päivitä tämä ohje, kun saat lämpötilan talletuksen ja tulostuksen toimimaan NodeJS:llä, MongoDb:llä ja expressillä.

Jotta lämpötilatietoja voitaisiin tallettaa ja tarkastella html-sivulla, niin asennetaan seuraavaksi LAMP-server (Linux, Apache, MySql, PHP) Raspberry PI:hin.

Kirjoita aluksi komento **sudo apt-get update**

Asenna seuraavaksi apache2 web-server komennolla **sudo apt-get install apache2 php5 libapache2-mod-php5 php5-mysql php5-cli –y**

Asennuksen jälkeen käynnistä apache server uudelleen komennolla **sudo /etc/init.d/apache2 restart**

Siirry html-hakemistoon komennolla **cd /var/www/html/**

Kopioi seuraava index.php tiedosto olemassa olevan html-tiedoston tilalle.

```
<?php
$hostname = 'localhost';
$username = 'pi_select';
$password = 'xxxxxxxxxx';

try {
    $dbh = new PDO("mysql:host=$hostname;dbname=measurements",
                                $username, $password);

    /*** The SQL SELECT statement ***/
    $sth = $dbh->prepare("
       SELECT  `dtg`, `temperature` FROM  `temperature`
    ");
    $sth->execute();

    /* Fetch all of the remaining rows in the result set */
    $result = $sth->fetchAll(PDO::FETCH_ASSOC);

    /*** close the database connection ***/
    $dbh = null;

}
catch(PDOException $e)
    {
        echo $e->getMessage();
    }

$json_data = json_encode($result);
?>

<!DOCTYPE html>
<meta charset="utf-8">
<style> /* set the CSS */

body { font: 12px Arial;}

path {
    stroke: steelblue;
    stroke-width: 2;
    fill: none;
}

.axis path,
.axis line {
    fill: none;
    stroke: grey;
    stroke-width: 1;
    shape-rendering: crispEdges;
}

</style>
<body>

<!-- load the d3.js library -->
```

```
<script src="http://d3js.org/d3.v3.min.js"></script>

<script>

// Set the dimensions of the canvas / graph
var margin = {top: 30, right: 20, bottom: 30, left: 50},
    width = 800 - margin.left - margin.right,
    height = 270 - margin.top - margin.bottom;

// Parse the date / time
var parseDate = d3.time.format("%Y-%m-%d %H:%M:%S").parse;

// Set the ranges
var x = d3.time.scale().range([0, width]);
var y = d3.scale.linear().range([height, 0]);

// Define the axes
var xAxis = d3.svg.axis().scale(x)
    .orient("bottom");

var yAxis = d3.svg.axis().scale(y)
    .orient("left").ticks(5);

// Define the line
var valueline = d3.svg.line()
    .x(function(d) { return x(d.dtg); })
    .y(function(d) { return y(d.temperature); });

// Adds the svg canvas
var svg = d3.select("body")
    .append("svg")
        .attr("width", width + margin.left + margin.right)
        .attr("height", height + margin.top + margin.bottom)
    .append("g")
        .attr("transform",
              "translate(" + margin.left + "," + margin.top + ")");

// Get the data
<?php echo "data=".$json_data.";" ?>
data.forEach(function(d) {
 d.dtg = parseDate(d.dtg);
 d.temperature = +d.temperature;
});

// Scale the range of the data
x.domain(d3.extent(data, function(d) { return d.dtg; }));
y.domain([0, d3.max(data, function(d) { return d.temperature; })]);

// Add the valueline path.
svg.append("path")
 .attr("d", valueline(data));
```

```
// Add the X Axis
svg.append("g")
  .attr("class", "x axis")
  .attr("transform", "translate(0," + height + ")")
  .call(xAxis);

// Add the Y Axis
svg.append("g")
  .attr("class", "y axis")
  .call(yAxis);

</script>
</body>
```