C.G.N. van Veen

HiSPARC

1 Weerstation Data

Inleiding We hebben ons weerstation werkend gekregen en krijgen nu data binnen op de computer. De weerdata komt zelfs via een wireless transmitter binnen. Je hebt zelf een behuizing gemaakt om je station tegen diverse weersomstandigheden te beschermen. In deze tutorial gaan we kijken hoe we onze metingen aan de database van HiSPARC kunnen toevoegen. We hebben dan op een goedkope wijze een eigen weerstation verkregen, waarmee we de efficiëntie van detectoren bij veranderende temperatuur en bijvoorbeeld het aantal showers als functie van de luchtdruk kunnen meten. Ons station is uitgerust met luchtdruk, luchtvochtigheid en temperatuur sensoren, maar is gemakkelijk uit te breiden met andere sensoren.

Benodigdheden

- Arduino software
- Python software
- PC van HiSPARC station
- Arduino weerstation

2 HiSPARC database

HiSPARC heeft een uitgebreide database waarin gegevens van kosmische straling wordt opgeslagen. Deze data is vrij voor iedereen om uit te lezen, te gebruiken voor onderzoek of om te leren omgaan met data en hoe je deze data kunt manipuleren met Python. Uitgebreide informatie over de HiSPARC database en hoe deze te benaderen is te vinden op `http://docs.hisparc.nl/publicdb/`. Hier vind je ook voorbeeld programma's in Python waarmee je zelf data kunt binnen halen. Wat wij willen doen is nu onze eigen weerdata koppelen aan de data van het HiSPARC meetstation, zodat deze data wordt opgeslagen in de HiSPARC database.

2.1 Data uitlezen en manipuleren

De HiSPARC database verwacht de data in een bepaald format en een bepaalde volgorde. Dat betekent dat we allereerst ons Arduino programma zo moeten maken dat de data er in de juiste volgorde uitkomt. In het onderstaande stukje code en de tabel zien we welke grootheden we in principe naar de HiSPARC database zouden kunnen sturen als we daar de sensoren van hadden aangesloten. De database van HiSPARC verwacht van het weerstation data in de volgende volgorde (zie tabel volgorde).

Grootheid (NL)	Grootheid (Engels in Python)	Eenheid
1. datum	Date	y-m-d
2. tijd	Time	h-min-s
3. temperatuur (detector)	`temp_inside`	°C
4. temperatuur (buiten)	`temp_outside`	°C
5. luchtvochtigheid (binnen)	`humidity_inside`	%
6. luchtvochtigheid (buiten)	`humidity_outside`	%
7. Luchtdruk	barometer	Pa
8. Windrichting	`wind_dir`	°
9. Windsnelheid	`wind_speed`	m/s
10. Zonne intensiteit	`solar_rad`	W/m^2
11. UV index	uv	(0-16)
12. Verdampingstranspiratie	evapotranspiration	mm
13. Hoeveelheid regen	`rain_rate`	mm
14. Gevoelstemperatuur	`heat_index`	°C
15. Dauwpunt	`dew_point`	°C
16. Gevoelstemperatuur (wind)	`wind_chill`	°C

Tabel 1 – Tabel met de 16 mogelijke grootheden die meegegeven kunnen worden aan de data van het weerstation. Een aantal van deze grootheden zoals gevoelstemperatuur kunnen berekend worden met behulp van de bekende grootheden als luchtvochtigheid en buitentemperatuur. Formules die daarvoor nodig zijn kunnen in de literatuur gevonden worden. `http://github.com/HiSPARC/weather/blob/master/doc/_static/Parameter_Manual.pdf`. Als de data geanalyseerd wordt kunnen deze grootheden alsnog berekend worden. Voor andere grootheden zoals bijvoorbeeld hoeveelheid neerslag moet een extra sensor worden aangesloten op de Arduino.

In de Python code houden we uit de tabel de Engelse grootheid aan. Als we een grootheid niet meten of kunnen uitrekenen dan wordt er een `"-999"` toegevoegd aan de datalijst. Daar moeten we in het Python programma ook rekening mee houden.

2.2 Test-procedure

Voordat we in de database gaan sleutelen moeten we eerst kijken of we de data in de juiste volgorde van de Arduino krijgen, alle sensoren hun data goed opsturen zodat de juiste data naar de HiSPARC database gezonden kan worden.

Allereerst gaan we zorgen dat ons Arduino programma de data zonder tekst en in de juiste volgorde (zoals in tabel 1) verstuurt. In code (1) hieronder zien we hoe dat moet.

```
// Code 1

// Programma geeft data van weerstation (Temperatuur (van een detector en van de
// buitenlucht), luchtvochtigheid (binnen -> -999 en buiten ) en luchtdruk in Pa.
// Gescheiden door komma's

#include <OneWire.h>  // to use data from multiple sensors send through one wire
#include <DallasTemperature.h> //library for ds18Bb20
```

```
#include <DHT.h>  // library for humidity sensors DHTxx (11, 21, 22)
#include <Wire.h>
#include <BMP085.h>

// code for digital Temperature sensor (ds18b20)
#define ONE_WIRE_BUS 3
// To database means we only use the temperature of one detector.
// Place the a temperature sensor in one of the detectors of the HiSPARC station.

// Setup DTH library with right sensor
DHT dht = DHT();

// Setup a oneWire instance to communicate with any OneWire devices in this
// case tempsensors
OneWire oneWire(ONE_WIRE_BUS);
// Pass our oneWire reference to Dallas Temperature.
DallasTemperature tsensors(&oneWire);
BMP085 dps = BMP085();      // Digital Pressure Sensor -> dps is accessing library
//needed for library of BMP085
long Temperature = 0, Pressure = 0, Altitude = 1000;

void setup(void) {
  // start serial port
  Serial.begin(9600);
  //Serial.println("HiSPARC Weather station measuring:");

  Wire.begin();
  dht.setup(5); // data pin 5 humidity
  dps.init(MODE_ULTRA_HIGHRES, 1000, true);  // 250 meters, true = using meter units
  tsensors.begin();
}

void loop(void) {
  // DTH22 Reading temperature or humidity takes about 250 milliseconds!
  // DTH22 Sensor readings may also be up to 2 seconds 'old' (its a very slow sensor)
  delay(dht.getMinimumSamplingPeriod());

  tsensors.requestTemperatures(); //Get temperature of detectors (one is default)
   for  (int deviceA = 0; deviceA < 1; deviceA++) {
     printTemp(deviceA);
  }
```

```
  float humidity = dht.getHumidity();
  float temperature = dht.getTemperature();

  // check if returns are valid, if they are nan (not a number)
  // then something went wrong!
  if (isnan(temperature) || isnan(humidity)) {
    continue;
  }

  else {
    Serial.print(temperature, 1); //Temperature outside
    Serial.print(",");
    Serial.print(humidity, 1); //humidity outside
    Serial.print(",");
  }
  dps.getTemperature(&Temperature);
  dps.getPressure(&Pressure);
  float pressure = Pressure/100;
  Serial.print(pressure); //luchtdruk in hPa
  Serial.println();
  delay(2000);
}

void printTemp(int adress) {
  float TempC = tsensors.getTempCByIndex(adress);
  String stringone = "Detector ";
  stringone += adress;
  Serial.print(" ");
  Serial.print(TempC,1); // print temperatuur van detector
  Serial.print(",");
}
```

2.3 Data gereed maken voor verzending naar database

De data komt als volgt: `24.6,24.4,37.8,1004.67` uit het Arduino programma. De eerste twee getallen zijn de temperatuur binnen en buiten, daarna volgt de luchtvochtigheid en de luchtdruk in hPa.

Nu schrijven we naast of in het python programma wat we al hadden geschreven in de handleiding: 'Wireless weerstation' (code 4 in die handleiding) een toevoeging, die de data zo manipuleert dat het in het format komt wat de HiSPARC database kan lezen.

Gebruik het onderstaande stukje code (2) in de python code die je al had. Uit je vorige code in py-

thon komt dus de Arduino data binnen. Die 'data' variabele bestaat dus uit vijf getallen. Die kun je naar de 'class' Measurement sturen met het volgende commando: `metingen = Measurement(data)` metingen is nu verwijzing naar alles wat in de class 'Measurement' met data wordt gedaan. Je kunt nu in de python code met `measurement.dew_point` bijvoorbeeld grootheden in de class 'Measurement' aanroepen.

```
# Code 2
# code om data het juiste format te geven. Ook het Dauwpunt wordt berekend.
# we werken met een Class.

# in de onderstaande Class komt de output van de Arduino binnen.

class Measurement(object):

    def __init__(self, output):

        # Datastore assumes date, time, temp_inside (detector),
        # temp_outside, humidity_inside, humidity_outside, barometer,
        # wind_dir, wind_speed, solar_rad, uv, evapotranspiration, rain_rate,
        # heat_index, dewpoint, wind_chill

        # Extracts variables from output, order of variables is important.
        # Set the order of variables in the Arduino Program.
        self.temp_inside, self.temp_outside, self.humidity_outside,
        self.barometer = output

        Tdew = self.dampdruk_calc(self.temp_outside, self.humidity_outside,
                                                        self.barometer)

        # As we do not measure these weather variables we set them to
        # '-999' If the weatherstation does measure these variables we
        # can add them to the list and extract them from the Arduino
        # output.

        self.wind_dir = '-999'
        self.humidity_inside = '-999'
        self.wind_speed = '-999'
        self.solar_rad  = '-999'
        self.uv = '-999'
        self.evapotranspiration = '-999'
        self.rain_rate = '-999'
        self.heat_index = '-999'
```

```
        self.dew_point = Tdew        # dew_point calculated.
        self.wind_chill = '-999'

        # Add timetstamp of measurement
        self.datetime = datetime.datetime.now()
        self.nanoseconds = 0


    def dampdruk_calc(self, Tout, Hum_out, baro):
        RH = Hum_out/100  #calculate relative humidity

        # Calculate vaporpressure, Dewpoint: Formula from Vantage Pro Davis instruments
        # This document can be found at:
        # http://github.com/HiSPARC/weather/blob/master/doc/_static/Parameter_Manual.pdf

        dampdruk = RH * 6.112 * numpy.exp((17.62 * Tout) / (Tout + 243.12))
        Numerator = 243.12 * numpy.log(dampdruk) - 440.1
        Denominator = 19.43 - numpy.log(dampdruk)
        Tdew = Numerator / Denominator
        Tdew = "%4.1f" %Tdew

        return Tdew
```

In de 'class Measurement' wordt ook het dauwpunt berekend. Als er andere afgeleide grootheden berekend moeten worden, kan dat gedaan worden door de '`def dampdruk_calc`' te kopiëren en en te plakken binnen de 'class Measurement'. Je kunt de '`def dampdruk_calc`' naam veranderen in bijvoorbeeld '`def wind_chill_calc`'. Voer nu de benodigde de berekening in met de grootheden die je gemeten hebt. Je kunt nu de '`wind_chill`' uit laten rekenen als je ook een sensor voor de windsnelheid hebt aangesloten. Voeg dan in de class Measurement in:

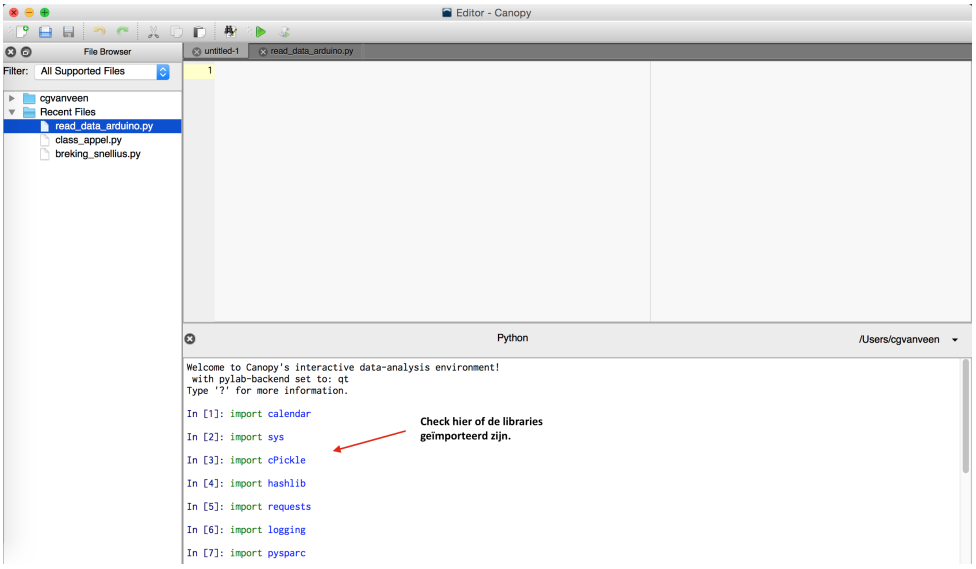
```
# Code 3a
# voorbeeld als de windspeed gemeten wordt kan wind_chill uitgerekend worden.

def wind_chill_calc(self, Tout, windspeed):

    # verdere berekeningen in het document Parameter_Manual.pdf, zie code 2.

    return wind_chill
```

Deze definitie berekent dan de gevoelstemperatuur. Deze kun je weer meegeven aan de lijst met data.



Figuur 3.1 – Venster van Canopy. Anaconda werkt op eenzelfde manier. Hier kun je in het interactieve venster invoeren 'import ...' en dan enter. Als de library niet is gevonden, geeft het programma een foutmelding.

```
# Code 3b
# voorbeeld als de windsnelheid bekend is

def __init__(self, output):
    #aanroepen functie
    windchill = self.wind_chill_calc(self.temp_outside, self.windspeed)

    #toekennen waarde
    self.wind_chill = windchill
```

Op dezelfde wijze kunnen er andere definities voor afgeleide grootheden binnen de class Measurement gemaakt worden.

De andere classes in het Python programma hoeven niet verandert te worden.

3 Data wegschrijven naar HiSPARC database: Windows/Mac.

3.1 Libraries van python installeren

Het gehele Python bestand waarmee je de data naar de HiSPARC database kunt schrijven is te vinden op: `https://github.com/HiSPARC/weather-arduino/tree/master/python` Om het geheel te laten werken zijn, moeten wel een aantal libraries geïmporteerd worden. Werk je in Anaconda onder windows, dan zijn de meeste libraries geïnstalleerd. Zo niet open dan een terminal. Ga naar 'Tools' en open een Terminal of Command prompt.

Nu kun je eventueel ontbrekende libraries installeren. Met het commando: 'pip freeze' kun je kijken welke libraries je al hebt.

Om te checken of libraries geïmporteerd zijn kun je in Anaconda in het interactieve venster invoeren: *import .....*. Zie Figuur 3.1 waarbij hetzelfde is gedaan met Canopy.

Als je geen foutmelding hebt dan is de library geïnstalleerd. In code 4 kun je zien welke libraries allemaal toegevoegd moeten zijn.

```
# Code 4
import serial
import datetime
import time
import logging
import calendar

import numpy

from pysparc import storage
```

Waarschijnlijk moet 'pysparc' nog geïnstalleerd worden. Open de terminal van Anaconda. Wat we gaan doen is een link naar de pysparc module van HiSPARC maken en deze gelijk installeren. type:

```
pip install http://github.com/HiSPARC/pysparc/archive/master.zip
```

Nu kun je `import pysparc` gebruiken, Zie Figuur 3.1.

3.2 Storage backup

Als de internet verbinding wegvalt dan willen we niet dat de data verloren gaat. We gebruiken daarom module *pysparc*. Deze gebruikt een database (Redis) om data op te slaan, totdat er weer een internetverbinding tot stand komt.

- Voor **windows** ga je naar:

```
http://github.com/rgl/redis/downloads
```

Download de versie van Redis, die je nodig hebt. Installeer Redis op je computer. Start nu de 'command prompt' als administrator. Run het volgende commando:

```
net start redis
```

Elke keer dat je de (Windows-)computer opnieuw opstart, moet wel Redis opnieuw opgestart worden.

- Voor de **Mac**: Als je 'Homebrew' geïnstalleerd hebt dan kun je eenvoudig het volgende in een terminal intypen: ('Homebrew' is eenvoudig te vinden met Google en dan te installeren op de Mac.

```
brew install redis
```

Heb je geen Homebrew. Open dan een terminal en type de volgende commando's in.

1. curl -O http://download.redis.io/redis-stable.tar.gz
2. tar -xvzf redis-stable.tar.gz
3. rm redis-stable.tar.gz
4. cd redis-stable
5. make
6. sudo make install
7. redis-server

De Redis server is nu geïnstalleerd en gestart.

3.3 Meten met het weerstation en data opsturen

Zorg dat je in Arduino het weerprogramma upload naar de Arduino (bijvoorbeeld met een USB kabel. Haal bij deze procedure het snoertje bij 'Rx' op de Arduino weg.) Bekijk in de Serial monitor of de de data correct wordt weergegeven. Sluit nu de UART-APC220 module aan op de Arduino en de PC.

Start Redis.

Open nu het Python programma (te vinden op `http://github.com/HiSPARC/weather-arduino`. `Datafromwirelesstodatabase.py`. Run het programma. *!Pas op!* Pas dit Python programma eventueel alleen aan in de 'class' die in deze handleiding benoemd is. Dus alleen in de class 'Measurement' kun je sensor waarden toevoegen.

Om te checken of het allemaal werkt draai je het programma in Anaconda. Vul zelf je HiSPARC station nummer in en het paswoord voor dat station. Het paswoord is te krijgen bij HiSPARC door te mailen naar `info@hisparc.nl`.

Door het Python script wordt alleen het dauwpunt geprint. Dit om te checken of er nog steeds gemeten wordt. Als het goed is zou het dauwpunt tussen een waarde van 0-10 moeten liggen. Dit printen kun je als commentaar uitschakelen.

Sluit het internet even af en kijk of na weer inschakelen van het internet weer data verzonden wordt. Als dat gebeurt is alles klaar voor gebruik. Werkt dat niet kijk, dan even of Redis opgestart is en run je het programma nog eens. Plaats nu je weerstation op het dak bij het station. Na een dag meten kun je bij de gegevens van het station ook de 'weer-grafiekjes' zien. Bekijk ze op `data.hisparc.nl`. Je eigen weerdata kun je nu altijd online uitlezen met dataretrieval tool. Zie de handleiding 'data retrieval' op `http://www.hisparc.nl/docent-student/lesmateriaal/informatie-pakket/`.