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 naast de metingen van kosmische straling met het HiSPARC station, ook de efficiëntie van detectoren bij veranderende temperatuur, aantal showers als functie van de luchtdruk kunnen meten. Ons station is uitgerust met luchtdruk en temperatuur, 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 meetstation wat je op school hebt, 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 mits 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)	tempInside	°C
4. temperatuur (buiten)	tempOutside	°C
5. luchtvochtigheid (binnen)	humidityInside	%
6. luchtvochtigheid (buiten)	humidityOutside	%
7. Luchtdruk	barometer	Pa
8. Windrichting	windDir	graden
9. Windsnelheid	windSpeed	m/s
10. Zonne intensiteit	SolarRad	W/m^2
11. UV index	UV	(0-16)
12. Verdampingstranspiratie	ET	mm
13. Hoeveelheid regen	rainRate	mm
14. Gevoelstemperatuur	heatIndex	°C
15. Dauwpunt	dewPoint	°C
16. Gevoelstemperatuur (wind)	windChill	°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 ultiple 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 right 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)) {
    Serial.println("Failed to read from DHT");
  }

  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.

Nu schrijven we naast het programma wat we al hadden 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 code die je al had.

```
# 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),
        # tempOutside, humidityInside, humidityOutside, barometer,
        # windDir, windSpeed, solarRad, UV, ET, rainRate heatIndex,
        # dewPoint, windChill

        # 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 is uitgerekend.
        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 Github/HiSPARC/weather/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 het dauwpunt berekend. Als er andere afgeleide grootheden berekend moeten worden, kan dat gedaan worden door de '`def dampdruk_calc`' te kopiëren en dan de berekening in te voeren met grootheden die je gemeten hebt. Je kunt bijvoorbeeld de 'windchill' uit laten rekenen als je ook een sensor voor de wind snelheid hebt aangesloten. Voeg dan tussen haakjes in:

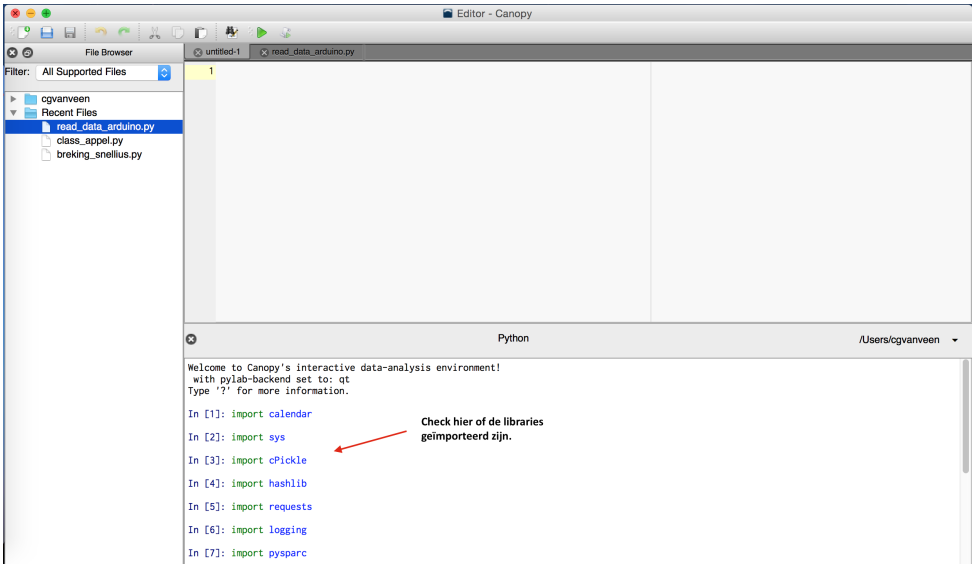
```
# Code 3

def windchill_calc(self, Tout, windspeed):

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

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 Canopy onder windows open dan een terminal. Ga naar 'Tools' en open een Canopy Terminal. Type in deze terminal:



Figuur 3.1 – Venster van Canopy.

```
enpkg setuptools
enpkg pip
```

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 Canopy in het onderste venster invoeren: *import .....*

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 numpy
import sys
import time
import hashlib
import requests
import logging
import calendar

import cPickle as pickle

from time import sleep
from pysparc import storage
```

Waarschijnlijk moet 'pysparc' geïnstalleerd worden. Open de terminal van Canopy. 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.

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:

```
\url{https://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 computer opnieuw opstart moet wel Redis draaien.
- voor de **Mac**: Als je 'Homebrew' geïnstalleerd hebt dan kun je eenvoudig het volgende in een terminal intypen:

```
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 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 dan 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 aan.

Start Redis.

Open nu het Python programma (te vinden op `http://github.com/HiSPARC/weather-arduino`. `Datafromwirelesstodatabase.py` Loop het programma even door. *!Pas op!* Pas alleen het programma alleen aan in de classes die in deze handleiding benoemd zijn.

Om te checken of het allemaal werkt draai je het programma in Canopy. Vul zelf je HiSPARC station in en het paswoord voor dat station. Het paswoord is te krijgen bij HiSPARC door te mailen naar `info@hisparc.nl`. 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 6-10 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. 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`.