Acquisizione di dati meteo per l'efficientamento energetico del CNR

Loredana Versienti[1] e Salvatore Di Cristofalo[2]

[1] ISTI-CNR, Via G. Moruzzi 1, 56124 Pisa, versienti@isti.cnr.it
[2] IAMC-CNR UOS di Capo Granitola, Via Del Mare 3, 91021 Torretta Granitola (TP), salvatore.dicristofalo@cnr.it

Abstract. Questo articolo analizza le possibili soluzioni tecniche per l'acquisizione di dati da stazioni meteo WheatherLink collegate ad Internet, al fine di creare una base di dati utile alla predisposizione di un piano di interventi per l'efficientamento energetico.

Keywords: Davis WheaterLink meteo station, Java, XML, Weewx

Introduzione

Il seguente documento riassume le possibili soluzioni investigate nell'acquisizione di dati meteorologici in tempo reale da stazioni meteo Davis WheatherLink [1]. Le stazioni meteo installate sul territorio italiano sono otto e sono ospitate nelle città di Bologna, Padova, Napoli, Pisa, Capo Granitola, Milano, Palermo, Roma. In figura 1 è mostrato un esempio di stazione meteo installata a Pisa.

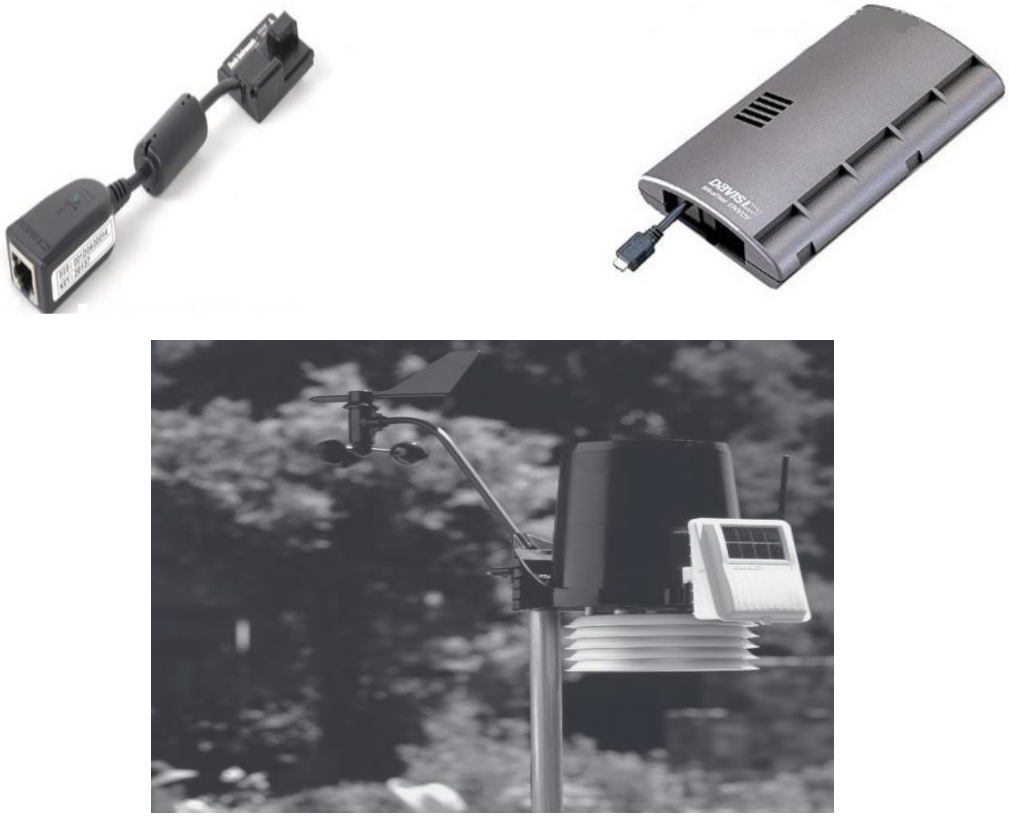

Figura 1: DataLogger, Weather Envoy, Stazione Meteo esterna

Stato dell'arte

In alcune applicazioni si presenta la necessità di recuperare informazioni aggiornate in modo automatizzato. Poichè le informazioni devono essere aggiornate, occorre ripetere nel tempo l'acquisizione dei dati, con una frequenza che varia in funzione dell'applicazione. Inoltre per le applicazioni che richiedono un aggiornamento in tempo reale occorre ideare una strategia per non sovraccaricare la fonte con ripetute richieste. Essendo un'attività ripetitiva, è importante che l'acquisizione venga automatizzata. Se le informazioni da acquisire sono disponibili in formato puro, ovvero prive di ridondanza e di formattazione, allora l'acquisizione è immediata. Viceversa se non si dispone di informazioni pure occorre processare il dato in ingresso per interpretarlo ed estrapolare l'informazione di interesse.

Le principali tecnologie di acquisizione si dividono in tecnologia *push* e tecnologia *pull*. La push descrive una tipologia di comunicazione in internet, dove la richiesta per ciascuna transazione è inizializzata dal publisher (il server centrale). Diversamente, nel modello pull la richiesta di trasmissione è inizializzata dal ricevente (il client).

I servizi push sono spesso basati su preferenze espresse in anticipo. Questo è chiamato modello pubblica/sottoscrivi. Un client può sottoscrivere svariati canali di informazione. Non appena diventa disponibile un nuovo contenuto all'interno di questi canali, il server esegue il push delle informazioni verso gli utenti (conferenze sincrone e messaggistica istantanea sono esempi tipici di servizi push). Una tipica implementazione del servizio push è il *long polling* che è una variazione del tradizionale polling per consentire l'emulazione della tecnologia push. Con il long polling il client richiede l'informazione al server, similmente a come farebbe con un *poll normale*. Se il server non dispone di informazioni per il client, invece di inviare una risposta vuota, trattiene la richiesta e aspetta che si rendano disponibili alcune informazioni. Una volta che l'informazione è disponibile (o dopo un certo timeout) viene inviata una risposta al client. Il client solitamente produrrà subito una nuova richiesta per il server, in modo che il server avrà sempre una richiesta in attesa che potrà essere utilizzata per inviare i dati in risposta a un evento.

L'utilizzo di una tecnologia di tipo push è possibile solo in seguito a un accordo esplicito tra publisher e client. Per ragioni puramente tecniche, è il publisher stesso che deve essere a conoscenza dei suoi client in modo da poterli avvertire non appena si rende disponibile il contenuto aggiornato; i servizi di tipo push possono garantire un alto livello di qualità e il perfetto sincronismo tra server e client. La tecnologia pull invece è una tecnica di comunicazione di rete che prevede una richiesta iniziale originata da un client, alla quale risponde un server. Le richieste pull sono il fondamento del network computing, dove molti client richiedono dati da server centralizzato.

Acquisizione dati con HTTP GET

Le stazioni Davis sono dotate di un data-logger visibile in alto a sinistra nella figura 1, che si collega ad Internet tramite un protocollo proprietario (WeatherLinkIP) il quale, in automatico e in modo trasparente all'utente, effettua l'upload dei dati registrati dalla stazione sul server del costruttore.

Mediante un account utente è possibile richiedere il trasferimento di un insieme di pacchetti dati (ogni pacchetto è relativo ad un timestamp) dal sito weatherlink.com, digitando in un browser la seguente riga:

*http://weatherlink.com/webdl.php?timestamp=0&user=*****&pass=*****&action=data*

Settando il timestamp pari a 0 si ottengono tutti i record memorizzati sul server fino alla data corrente. Il server weatherlink.com può immagazzinare fino a 2 anni di dati.

Le informazioni contenute in un pacchetto sono organizzate in un record di 52 bytes, strutturato secondo i seguenti campi, come riportato nel manuale [2] fornitoci dalla Davis Instruments Corp:

Field	Offset	Size	Dash Value	Explanation
Date Stamp	0	2	Not applicable	These 16 bits hold the date that the archive was written in the following format: Year (7 bits) | Month (4 bits) | Day (5 bits) or: day + month*32 + (year-2000)*512)
Time Stamp	2	2	Not applicable	Time on the Vantage that the archive record was written: (Hour * 100) + minute.
Outside Temperature	4	2	32767	Either the Average Outside Temperature, or the Final Outside Temperature over the archive period. Units are (°F / 10)
High Out Temperature	6	2	-32768	Highest Outside Temp over the archive period.
Low Out Temperature	8	2	32767	Lowest Outside Temp over the archive period.
Rainfall	10	2	0	Number of rain clicks over the archive period
High Rain Rate	12	2	0	Highest rain rate over the archive period, or the rate shown on the console at the end of the period if there was no rain. Units are (rain clicks / hour)
Barometer	14	2	0	Barometer reading at the end of the archive period. Units are (in Hg / 1000).
Solar Radiation	16	2	32767	Average Solar Rad over the archive period. Units are (Watts / m^2)
Number of Wind Samples	18	2	0	Number of packets containing wind speed data received from the ISS or wireless anemometer.
Inside Temperature	20	2	32767	Either the Average Inside Temperature, or the Final Inside Temperature over the archive period. Units are (°F / 10)
Inside Humidity	22	1	255	Inside Humidity at the end of the archive period
Outside Humidity	23	1	255	Outside Humidity at the end of the archive period
Average Wind Speed	24	1	255	Average Wind Speed over the archive interval. Units are (MPH)
High Wind Speed	25	1	0	Highest Wind Speed over the archive interval. Units are (MPH)
Direction of Hi Wind Speed	26	1	32767	Direction code of the High Wind speed. 0 = N, 1 = NNE, 2 = NE, … 14 = NW, 15 = NNW, 255 = Dashed

Prevailing Wind Direction	27	1	32767	Prevailing or Dominant Wind Direction code. 0 = N, = NNE, 2 = NE, … 14 = NW, 15 = NNW, 1 255 = Dashed Firmware before July 8, 2001 does not report direction code 255
Average UV Index	28	1	255	Average UV Index. Units are (UV Index / 10)
ET	29	1	0	ET accumulated over the last hour. Only records "on the hour" will have a non-zero value. Units are (in / 1000)
High Solar Radiation	30	2	0	Highest Solar Rad value over the archive period. Units are (Watts / m^2)
High UV Index	32	1	0	Highest UV Index value over the archive period. Units are (Watts / m^2)
Forecast Rule	33	1	193	Weather forecast rule at the end of the archive period.
Leaf Temperature	34	2	255	2 Leaf Temperature values. Units are (°F + 90)
Leaf Wetnesses	36	2	255	2 Leaf Wetness values. Range is 0 – 15
Soil Temperatures	38	4	255	4 Soil Temperatures. Units are (°F + 90)
Download Record Type	42	1		0xFF = Rev A, 0x00 = Rev B archive record
Extra Humidities	43	2	255	2 Extra Humidity values
Extra Temperatures	45	3	32767	Extra Temperature values. Units are (°F + 3 90)
Soil Moistures	48	4	255	4 Soil Moisture values. Units are (cb)

Figura 2: struttura di un pacchetto dati inviato dalla stazione

Come risultato di questa richiesta abbiamo prelevato un file binario che è stato decodificato secondo il formato descritto dalla tabella di figura 2. Il linguaggio utilizzato per la decodifica è JAVA7 [3] mentre come ambiente di lavoro abbiamo scelto l'IDE ECLIPSE [4], installato su una macchina Windows.
Per ogni stazione l'attività è consistita nell'interrogare il server, decodificare il file e memorizzare i dati ottenuti nel formato tabellare visibile in figura 3.

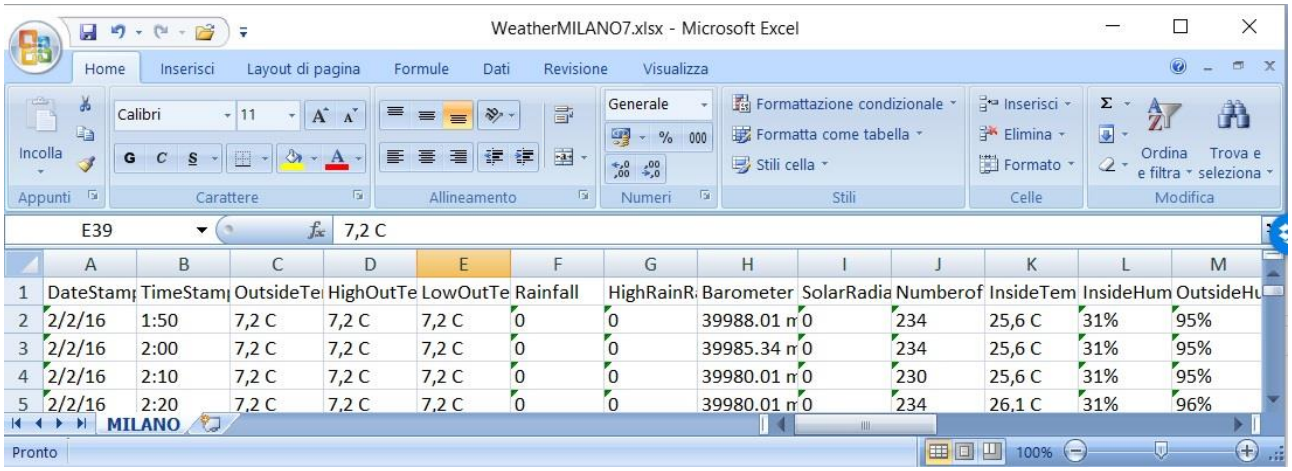

Figura 3: decodifica dei dati inviati dalla stazione di Milano

Analizzando i dati di tutte e otto le stazioni meteo, ci siamo resi conto che questo approccio non faceva al nostro caso, in quanto i pacchetti prelevati dal server weatherlink.com non sono aggiornati fino al timestamp corrente, ma hanno un ritardo di circa un'ora. Ciò rendeva impraticabile l'utilizzo

di questa soluzione in quando il nostro obbiettivo è quello di fare polling ad intervalli temporali non superiori ai cinque minuti.

Approfondendo la conoscenza della stazione ci siamo resi conto che le Davis WheaterLink installate sono dotate di un sottosistema di *datalogger,* il quale è costituito da una memoria di 3.9 MB e da una scheda Ethernet. Si è palesata così l'idea di andare ad interrogare direttamente questo sottosistema per il quale il produttore Davis fornisce le specifiche del protocollo utilizzato dal datalogger per le comunicazioni di rete. Tale protocollo si appoggia sulla ben nota pila protocollare TCP/IP.

Sebbene l'idea di implementare un protocollo di comunicazione ci avrebbe permesso di superare qualsiasi "vincolo" di reperimento dei dati imposti dal costruttore (come quello di passare necessariamente attraverso un server weatherlink.com), ciò sarebbe stato troppo dispendioso in termini di tempo e manpower, per cui abbiamo abbandonato l'idea.

Acquisizione dati in formato XML

Un ulteriore modalità offerta dal server weatherlink.com è l'acquisizione di dati tramite il protocollo HTTP di un file XML (tecnologia di tipo pull).
Con l'ausilio di un browser e con le credenziali ottenute durante la registrazione della stazione meteo, si digita l'URI:

*http://www.weatherlink.com/xml.php?user=*****&pass=*****

ed in risposta si ottiene:

```
<current_observation
version="1.0"xsi:noNamespaceSchemaLocation="http://www.weather.gov/view/
current_observation.xsd">
<credit>Davis Instruments Corp.</credit>
<credit_URL>http://www.davisnet.com</credit_URL>
<image>
  <url>http://www.weatherlink.com/images/Logo_Davis_reflxblu.jpg</url>
  <title>DavisWeatherLink</title>
  <link>http://www.weatherlink.com</link>
</image>
<suggested_pickup>15 minutes after the hour</suggested_pickup>
<suggested_pickup_period>60</suggested_pickup_period>
<dewpoint_c>9.4</dewpoint_c>
<dewpoint_f>49</dewpoint_f>
<dewpoint_string>49 F (9.4 C)</dewpoint_string>
<heat_index_c>13.9</heat_index_c>
<heat_index_f>57</heat_index_f>
<heat_index_string>57 F (13.9 C)</heat_index_string>
<latitude>43.719657402243</latitude>
<location>Pisa, Toscana,Italia</location>
<longitude>10.422307836423</longitude>
<observation_time>Last Updated on Feb 28 2016, 9:28 pm CET</observation_time>
<observation_time_rfc822>Sun, 28 Feb 2016 21:28:26 +0100</observation_time_rfc822>
<pressure_in>29.316</pressure_in>
<pressure_mb>992.7</pressure_mb>
<pressure_string>992.7 mb</pressure_string>
<relative_humidity>72</relative_humidity>
<station_id>pi01e</station_id>
<temp_c>14.2</temp_c>
<temp_f>57.6</temp_f>
<temperature_string>57.6 F (14.2 C)</temperature_string>
```

Figura 4: dati meteo in formato XML della stazione di Pisa

L'approccio percorribile per acquisire i dati in un database relazionale è quello di effettuare il parsing del file XML e se si utilizza Java come linguaggio di sviluppo, la libreria che supporta questa procedura è StAX Parser che la troviamo già inclusa nel JRE.
Anche in questo caso per evitare sforzi inutili, pur avendo le competenze per lo sviluppo, debugging e testing del codice abbiamo approfondire lo studio di una ulteriore soluzione rappresentata dal framework open-source *Weewx* [5].

Weewx

Weewx è software open source scritto in Python che interagisce direttamente con una stazione meteo per produrre grafici, report e pagine html. Supporta diverse stazioni meteo tra cui anche le stazioni Davis di nostro interesse. Può essere eseguito su diverse piattaforme PC/Linux, compresi economici sistemi come quello del Raspberry/Linux. Alcune caratteristiche chiave sono:

- Compatibilità con diverse stazioni meteo (AcuRite, Davis, LaCrosse, Oregon Scientific, ...)
- Possibilità di trasferimento dati su un sito FTP
- Supporto nativo per diversi sistemi di unità di misura
- Supporto per i database SqlLite e MySQL

Il software risulta essere molto flessibile in quando permette di personalizzare non solo il repository dove memorizzare i dati meteo rilevati, ma anche di formattare i dati in diverse unità di misura, ad esempio esprimere la temperatura in gradi Celsius piuttosto che in Fahrenheit.

L'attività di acquisizione e storicizzazione dei dati da parte del software avviene in automatico ed è trasparente all'utente, il quale deve soltanto scegliere, in fase di configurazione, uno dei due database proposti.

Per la scelta del database ci siamo orientati verso la soluzione MySQL, in quanto permette da una parte, al servizio weewx di memorizzare i dati in tempo reale (ad esempio ogni cinque minuti), dall'altra ad una applicazione in esecuzione su una macchina diversa di interrogare il Server MySQL per richiedere in modo affidabile dati con una cadenza maggiore od uguale a quella preimpostata in weewx (weewx può essere eseguito in background come un qualsiasi servizio Linux):

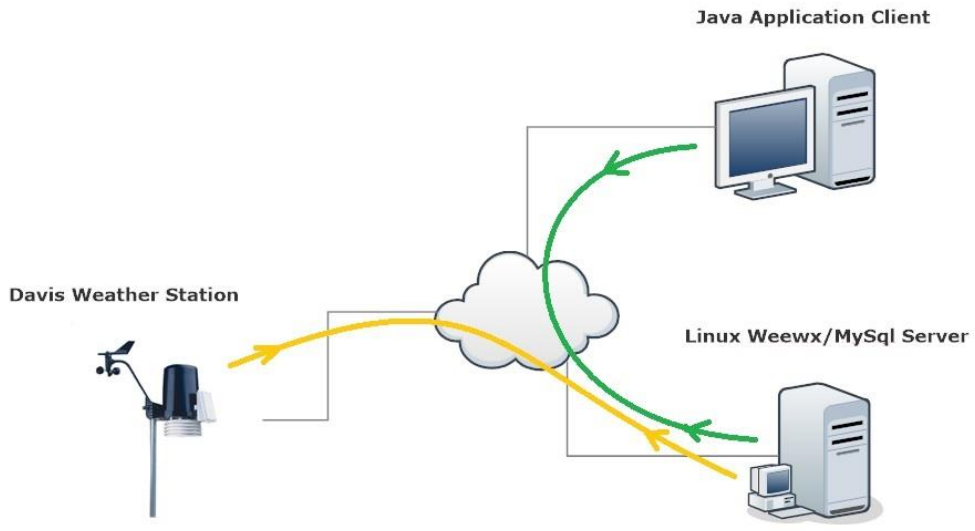

Figura 5: architettura del sistema

Dopo molti tentativi (data la scarsa documentazione in merito), è emerso che la stazione meteo è "stateful", cioè si comporta in base alle chiamate ricevute: se a interrogarla sono due processi diversi, allora le risposte che ciascun processo riceve sono condizionate dalle interazioni che l'altro processo ha nel frattempo avuto con la stazione. Questo perchè il datalogger accetta una sola richiesta TCP/IP e non gestisce la concorrenza tra processi. Ne consegue che per il corretto funzionamento della nostra architettura *solo* il servizio weewx deve poter accedere per cui abbiamo dovuto configurare le stazioni in modo che non inviassero più i dati al sito del produttore.

Da notare che l'architettura raffigurata in fig. 5 è scalare cioè, date più stazioni meteo, possiamo avviare più servizi weewx su una stessa macchina e memorizzare i dati in database differenti gestiti dallo stesso DB Managment System. In questo modo una applicazione front-end può richiedere al server dati meteorologici provenienti da stazioni differenti.

L'installazione del framework weewx è molto lineare poichè il software può essere scaricato sotto forma di archivio "rpm" e non di meno lo è la configurazione e l'avvio del servizio. Una volta installato il software è necessario editare il file di configurazione */etc/weewx/weewx.conf* specificando:

- il tipo di stazione (Vantage)
- il tipo di connessione (ethernet)
- l'indirizzo IP e la porta della stazione
- il driver
- il tipo di database
- il nome e la locazione del DB
- l'intervallo temporale di interrogazione.

Seguono due immagini che mostrano due sezioni del file di configurazione:

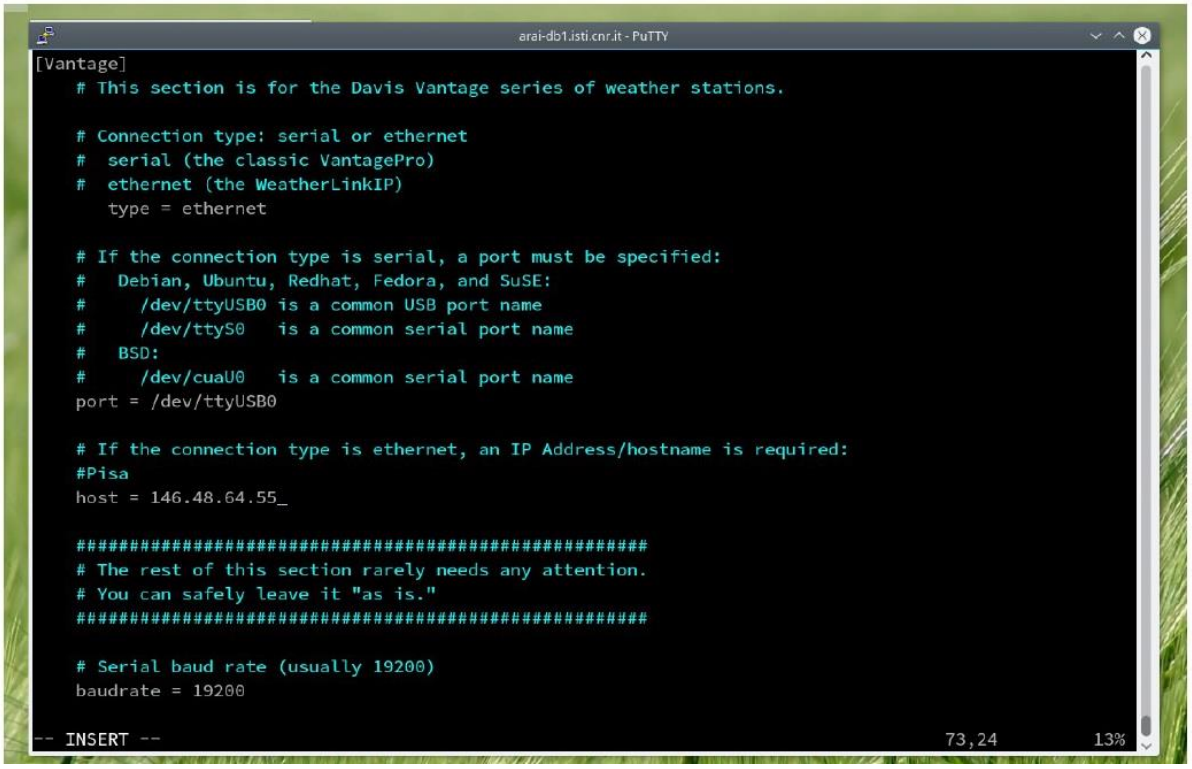

Figura 6: sezione [Vantage]

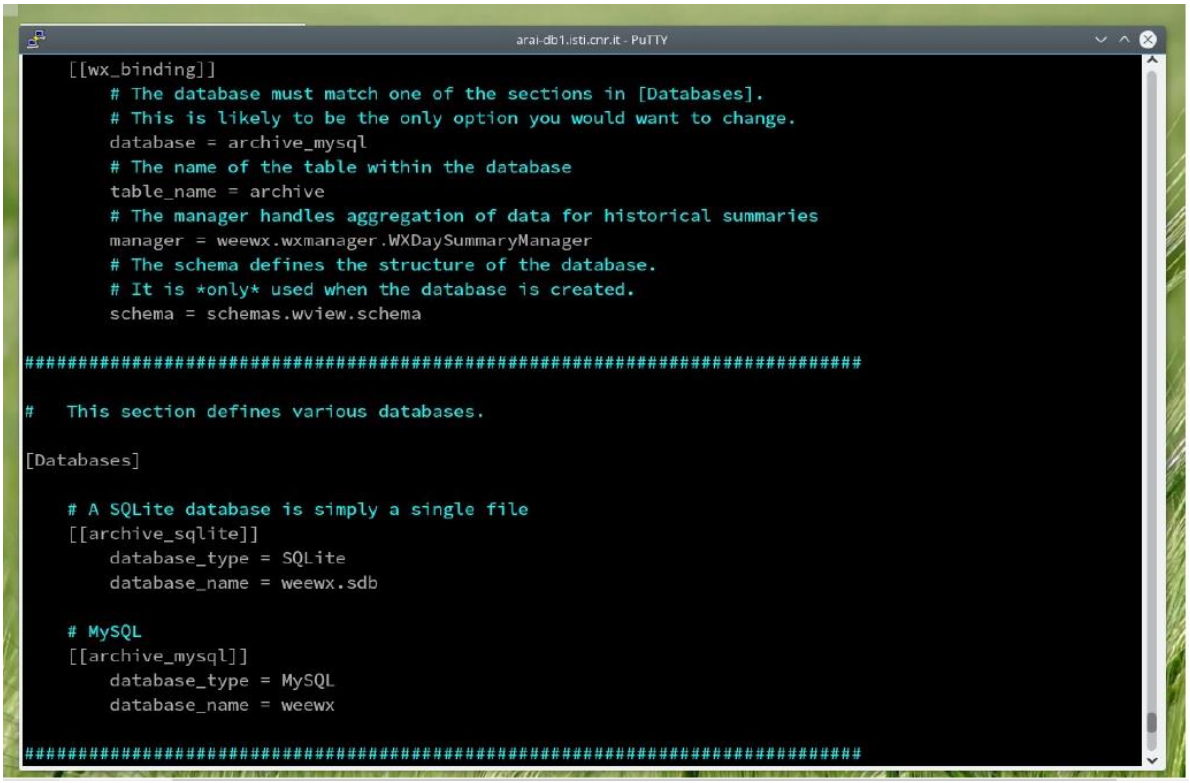

Figura 7: sezione di configurazione del DB MySQL

Per attivare il servizio weewx è sufficiente digitare il comando:

bash# *sudo /etc/init.d/weewx start*

Il servizio, una volta attivo, inizia ad interrogare la stazione con la cadenza temporale specificata nel file di configurazione, prelevando i dati meteo e memorizzandoli nel DB. Per fermare il servizio invece è necessario digitare il comando:

bash# *sudo /etc/init.d/weewx stop*

In figura 8 sono riportati i dati acquisiti nel database relazionale.

dateTime	usUnits	interval	barometer	pressure	altimeter	inTemp	outTemp	inHumi...	outHum...	windSp...	windDir	windGust
1456071...	1	1	30.244			78.5	54.3	35	69	0		1
1456071...	1	1	30.245			78.6	54.3	35	69	0		1
1456071...	1	1	30.244			78.6	54.3	35	69	1	22.5	1
1456072...	1	1	30.243			78.6	54.3	35	69	1	22.5	2
1456072...	1	1	30.242			78.6	54.3	35	69	1	22.5	2
1456072...	1	1	30.242			78.6	54.3	35	70	1	22.5	1
1456072...	1	1	30.241			78.5	54.3	35	69	2	22.5	3
1456072...	1	1	30.242			78.5	54.3	35	69	1	22.5	2
1456072...	1	1	30.242			78.6	54.2	36	69	2	22.5	3
1456072...	1	1	30.242			78.6	54.2	36	69	2	22.5	3
1456072...	1	1	30.242			78.6	54.2	36	69	1	22.5	4
1456072...	1	1	30.243			78.6	54.2	36	69	2	22.5	3
1456072...	1	1	30.242			78.6	54.1	36	69	2	22.5	3
1456072...	1	1	30.243			78.6	54.1	36	69	1	22.5	2
1456072...	1	1	30.243			78.6	54.1	36	70	1	22.5	1

Figura 8: dati meteo sul database relazionale

Interfaccia di consultazione

Oltre all'acquisizione dei dati dalle stazioni meteo, che avviene in modo automatico e trasparente all'utente, il sistema dovrà fornire anche un'interfaccia di consultazione. L'interazione tra utente finale e sistema dovrà avvenire esclusivamente tramite l'interfaccia. Il sistema deve prevede almeno due modalità di consultazione dei dati:

- Modalità "download" di dati raw al fine di mettere a disposizione informazioni meteo a chiunque ne faccia richiesta, per qualsivoglia analisi.
- Modalità visuale dei dati per consultazioni in tempo reale delle condizioni meteo (valori istantanei, grafici statistici, ecc.)

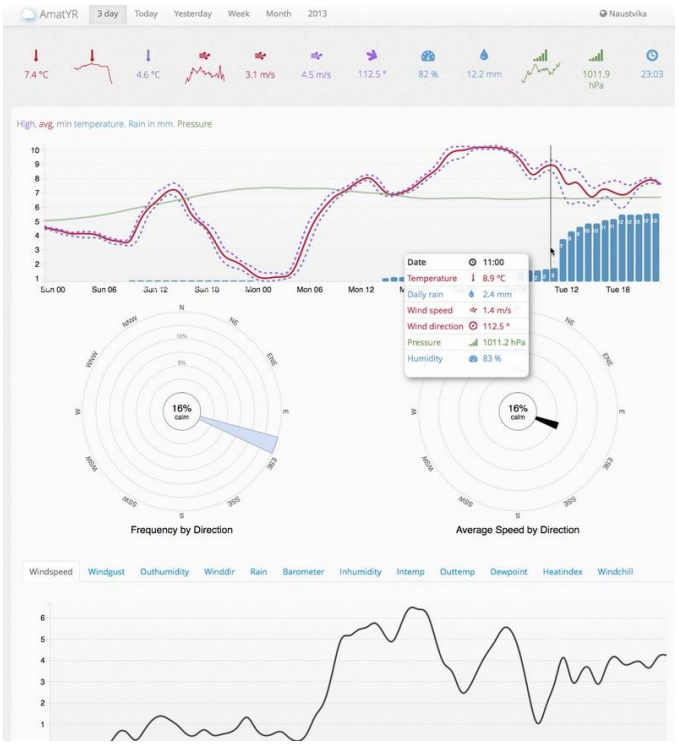

Figura 9: consultazione grafica di dati meteo

Conclusioni

Le diverse possibilità esplorate ci hanno permesso di approfondire fino in fondo le caratteristiche di un prodotto commerciale, che per molti aspetti risulta essere di difficile comprensione, forse proprio per il fatto che volutamente si vogliono preservare le caratteristiche architetturali/software di un prodotto proprietario. Riteniamo che il software open source Weewx possa essere approfondito come oggetto di studio e di possibile soluzione al nostro obbiettivo: fare il polling della stazione meteo con un intervallo di tempo prefissato.

Un aspetto di non secondaria importanza è la memorizzazione e storicizzazione dei dati meteorologici in modo tale da rendere la base di dati uno strumento efficace per migliorare il monitoraggio dei consumi energetici delle utenze del CNR. Queste informazioni sono indispensabili per effettuare diagnosi energetiche approfondite delle utenze e per predisporre un piano di interventi per l'efficientamento energetico.

Ringraziamenti

Questo lavoro è stato finanziato dal Dipartimento DIITET del CNR, fa parte del più ampio progetto "Energia da Fonti Rinnovabili e ICT per la Sostenibilità Energetica".

Riferimenti

[1] http://www.davisnet.com/weather/
[2] http://www.davisnet.com/support/weather/download/VantageSerialProtocolDocs_v261.pdf
[3] https://www.oracle.com/java/index.html
[4] https://eclipse.org/
[5] http://www.weewx.com