Stazione Meteorologica

Progetto per il corso di
Sistemi embedded

di
Gioele Masini
matricola: 657158

Corso di
Ingegneria e Scienze Informatiche
anno 2014/2015

Indice

1 Introduzione

Il progetto consiste nella realizzazione di una stazione meteorologica in grado di effettuare rilevazioni autonome sull'ambiente circostante e immagazzinare i dati raccolti. Il sistema dispone inoltre di adeguate interfacce web per la richiesta di tali dati da parte di applicazioni terze e per la visualizzazione da parte di utenti desktop e mobile. In quest'ultimo caso i dati vengono rielaborati e presentati tramite grafici.

L'obiettivo del progetto è quello di fornire un unico centro di raccolta dati attorno al quale possa essere strutturato un ecosistema completo per una smart-home o per l'ambito industriale.

Più dettagliatamente le rilevazioni sono affidate ad un Arduino il quale si occupa di recuperare i dati dei sensori collegati rielaborando i segnali in formato leggibile e mostrandoli tramite un display LCD. Ad intervalli costanti si occupa inoltre di trasmetterli sull'interfaccia con la quale è connesso ad un Raspberry Pi. Questi mantiene uno script in ascolto leggendo di volta in volta i nuovi dati e registrandoli in un database SQLite. Si occupa inoltre di fornire un'interfaccia web che permette il recupero dei dati raw in formato JSON, il download in formato CSV e la visualizzazione tramite grafici. I due dispositivi sono connessi tra loro tramite interfaccia seriale (USB) mentre il Raspberry Pi deve essere connesso alla rete locale via Ethernet o wifi (tramite dispositivo esterno) per rendere disponibile l'interfaccia web agli altri dispositivi connessi.

1.1 Tecnologie e materiali utilizzati

Software utilizzati per la progettazione e lo sviluppo:

- Arduino IDE
- Notepad++
- Fritzing
- Visual Paradigm 11

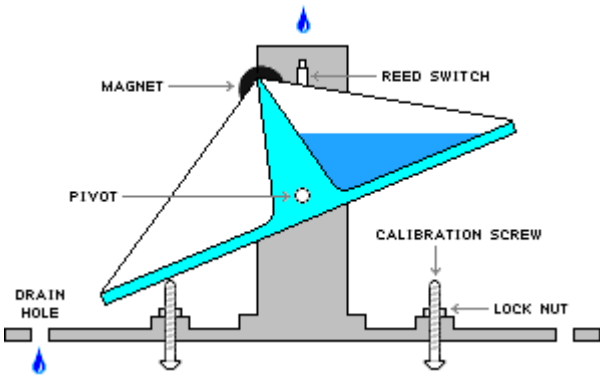
Componenti hardware del sistema:

- Arduino Uno Rev 3
- Raspberry Pi 1, model B
- Sensore di temperatura LM35
- Fotoresistenza
- Anemometro
- Pluviometro
- Schermo LCD 16x2, compatibile con driver Hitachi HD44780
- Potenziometro lineare 10 kΩ
- Resistenze:
  - 1x 470 Ω
  - 1x 1 kΩ
- Cavi per cablaggi
- Cavo USB type B

- Cavo Ethernet
- Breadboard

1.2 Dettagli hardware

- *Anemometro* (sensore di rilevamento velocità del vento): l'anemometro utilizzato è il classico modello costituito da tre coppette ruotanti su un perno centrale. Come da specifiche fornite dal produttore avviene una momentanea chiusura del contatto quando internamente il magnete, ruotando, passa sopra all'interruttore. Tale chiusura può essere rilevata da un pin di input digitale tramite interrupt opportunamente configurato. Con il vento a velocità di 2.4 km/h il sensore effettua una chiusura al secondo.
- *Pluviometro* (sensore di rilevamento pioggia): il pluviometro utilizzato è della tipologia "a doppia vaschetta basculante". In accordo con il datasheet del produttore questi effettua una momentanea chiusura del contatto ogni 0.2794 mm di pioggia rilevabile da un pin di input digitale tramite interrupt opportunamente configurato. La chiusura è provocata dall'oscillazione delle due vaschette che muove il magnete sopra all'interruttore interno.



Pluviometro a doppia vaschetta basculante

2 Scelte architetturali e specifiche tecniche

Le scelte operate in tutti gli ambiti del progetto sono state orientate all'efficienza nell'uso delle risorse, e alla manutenibilità, estensibilità e riusabilità del codice. Tali qualità sono state scelte considerando l'hardware e l'ambito di utilizzo del sistema. Si ricavano quindi due requisiti principali:

- **Efficienza nell'uso delle risorse**: i componenti hardware in ambito embedded dispongono di risorse molto limitate ed è necessario disporne ottimamente per sviluppare un sistema rapido nella risposta;
- **Estensibilità del sistema**: un sistema di lettura dati centralizzato per smart-home richiede una facile estensibilità e modularità per poter essere personalizzato con le componenti che meglio si adattano alle varie situazioni e necessità.

Nello specifico il database scelto per la registrazione dei dati è SQLite in quanto più leggero e adatto ad hardware meno potenti rispetto ai database tradizionali. Inoltre, essendo utilizzato solo per la registrazione e il recupero dei dati, non si è resa necessaria la potenza e flessibilità dell'SQL tradizionale. L'unico neo di tale tecnologia si è rivelato la mancanza del supporto al tipo di dato "datetime", risolto poi convertendo il valore in millisecondi e registrandolo come intero. Per l'ottimizzazione dei dati, necessaria a ridurre i tempi di caricamento dei dati da interfaccia web, sono stati utilizzati due file esterni per ogni anno (uno per i dati mensili e uno per quelli annuali) nei quali vengono salvati i relativi dati serializzati. La scelta dell'utilizzo di un file esterno è stata dettata dalla volontà di mantenere separati i dati registrati dai sensori e quelli ridondanti calcolati dal sistema. E' così possibile effettuare modifiche al codice di ottimizzazione senza dover modificare la struttura del database e salvare copie di backup del database con i soli dati utili. Quest'ultimo è facilitato anche dalla struttura dei database di SQLite, composti da un solo file. In caso di ripristino di un backup sarà quindi sufficiente effettuare la sola ottimizzazione quando si utilizzerà la visualizzazione web.

L'interfaccia di condivisione dei dati con l'esterno è completamente web per semplificare tali operazioni e renderle accessibili da qualsiasi dispositivo connesso. E' possibile richiedere i dati raw in formato JSON o visualizzarli tramite interfaccia HTML e Javascript organizzati in grafici, suddivisi per dati del giorno, del mese e dell'anno. Il formato JSON è stato utilizzato anche nell'invio dei dati da Arduino permettendo così la scrittura di uno script in ascolto flessibile in quanto indipendente dai dati concretamente ricevuti.

Sia il software su Arduino che su Raspberry è stato scritto utilizzando il paradigma ad oggetti, ove possibile.

2.1 Software Arduino

Lo sketch scritto per Arduino si occupa di leggere i dati dai sensori, elaborarli secondo le unità di misura scelte, aggiornare lo schermo LCD e inviare i risultati sull'interfaccia seriale ad intervalli costanti, temporizzati tramite l'uso del timer1 di Arduino configurato tramite la libreria *TimerOne*.
I seguenti punti descrivono nel dettaglio la struttura, le tecnologie utilizzate e i compiti del software installato su Arduino:

- **Lettura dati e aggiornamento LCD** (*C++*)

  Il programma istanzia un oggetto per ogni sensore utilizzato, implementando la classe astratta Sensor. Tali oggetti vengono poi registrati insieme all'LCD all'interno della classe WeatherStation la quale contiene i metodi pubblici per leggere i dati da tutti i sensori, per aggiornare i valori sullo schermo e per inviarli sull'interfaccia seriale. L'istanza di

WeatherStation si occupa internamente di calcolare la media dei valori ottenuti da ogni lettura che viene azzerata ad ogni invio dei dati su interfaccia seriale.

- **Invio dati** (*C++*)

  L'invio dei dati su interfaccia seriale avviene ad intervalli costanti, uguali o multipli dell'aggiornamento dell'LCD. I dati inviati sono il risultato delle rilevazioni effettuate nell'intervallo tra un invio e il successivo, in base alla tipologia di valore viene effettuata la somma o la media di tutte le rilevazioni. I dati vengono inviati in JSON usando la libreria JsonArduino che, allocando la memoria utilizzata staticamente, permette di avere ottime prestazioni e un uso della memoria predittivo. I dati sono inviati nella forma: *"nomeValore": valore*.

2.2 Software Raspberry Pi

I seguenti punti descrivono nel dettaglio la struttura, le tecnologie utilizzate e i compiti del software installato sul Raspberry Pi:

- **Registrazione dati** (*Python*)

  La lettura e il salvataggio dei dati inviati da Arduino su interfaccia seriale vengono eseguiti da uno script Python in esecuzione in background, ciclicamente in ascolto sull'USB. Dopo ogni lettura si occupa anche di calcolare ed associare ai dati la data e l'ora della registrazione, in millisecondi.

  Per lo script è stato scelto un linguaggio interpretato in quanto non si necessita di un programma particolarmente complesso o performante e si è preferito quindi evitare di dover compilare il codice in favore di una maggior flessibilità, un minor tempo di sviluppo e una gestione più semplice e flessibile del codice. Python inoltre fornisce una libreria semplice e funzionale per la lettura dei dati da interfaccia seriale.

  Lo script si occupa di creare, se non presente, una tabella nel database inserendo tante colonne di tipo REAL quanti sono i valori ricevuti da Arduino nominandole come le stringhe *"nomeValore"* dell'oggetto JSON ricevuto.

- **Conservazione dati** (*Database SQLite3*)

  Tutti i valori letti dallo script di registrazione dati vengono salvati in un database apposito insieme alla data e ora di registrazione. Il database è costituito da una tabella per anno solare create autonomamente dallo script Python e contenenti tutti i valori registrati dal sistema. Le tabelle sono nominate con il numero dell'anno preceduto dal prefisso "*data_*".

  Per diminuire i dati inviati come risposta alle richieste tramite interfaccia web vengono inoltre generati dei file di cache persistenti nella cartella /data/cache/. Per ogni anno ottimizzato vengono salvati due file, uno con risoluzione mensile e uno annuale. Tali dati vengono creati su richiesta dell'utente da uno script PHP che calcola le medie necessarie per ottenere 1 valore all'ora per la vista mensile e 1 al giorno per quella annuale.

- **Interfaccia Web** (*HTML5, Javascript e PHP*)

  L'interfaccia verso l'esterno è completamente web, permette agli utenti di visualizzare i dati organizzati in grafici per giorno, mese e anno della data scelta. E' possibile inoltre richiedere tali dati in formato JSON oppure scaricarli in formato CSV.

  Per la creazione dell'interfaccia sono stati utilizzati tutti gli strumenti classici della programmazione web. Lato server è stato scelto PHP 5 in quanto, rispetto alla versione precedente, fornisce molte delle proprie librerie riscritte secondo il paradigma ad oggetti e

tra di esse ne è presente una apposita per la gestione dei database SQLite3. Sono state inoltre scritte diverse ulteriori classi per fornire le funzionalità di connessione e lettura dei dati dal database, l'ottimizzazione dei valori, il download in formato CSV ed è stata estesa la classe DateTime per supportare la conversione in millisecondi equivalenti (utilizzati nel database e su Javascript) e altre funzionalità utili. Lato client invece è stata utilizzata la libreria Javascript "*Google Charts*" per il rendering dei dati sotto forma di grafici e gli standard HTML5 e CSS3 per strutturare le varie pagine web. E' stata anche abbozzata la visualizzazione da mobile usando una libreria PHP per distinguere i client desktop dagli smartphone inviando i relativi fogli di stile.

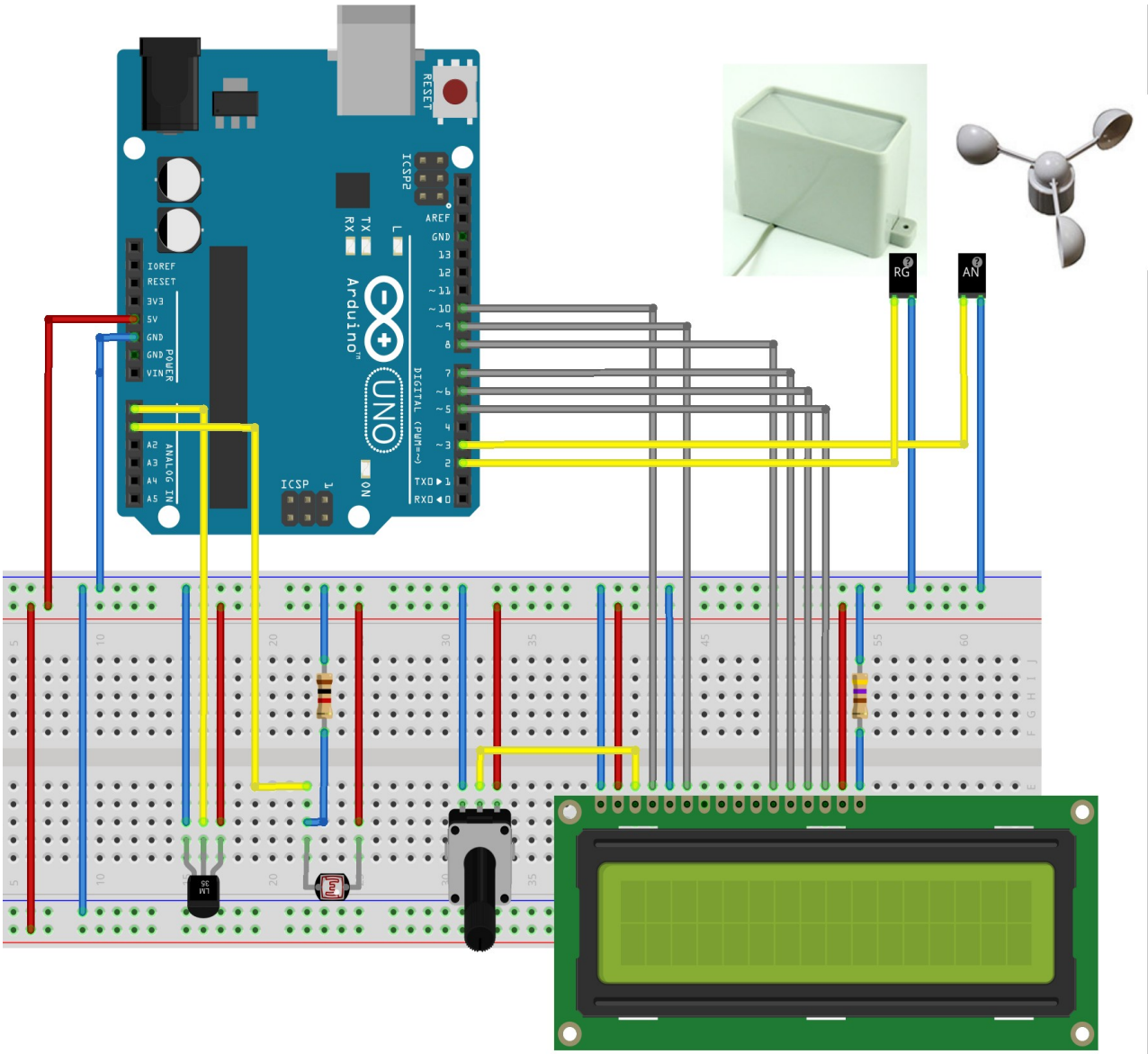
3 Struttura

Questa sezione riporta la descrizione della struttura hardware e software del sistema.

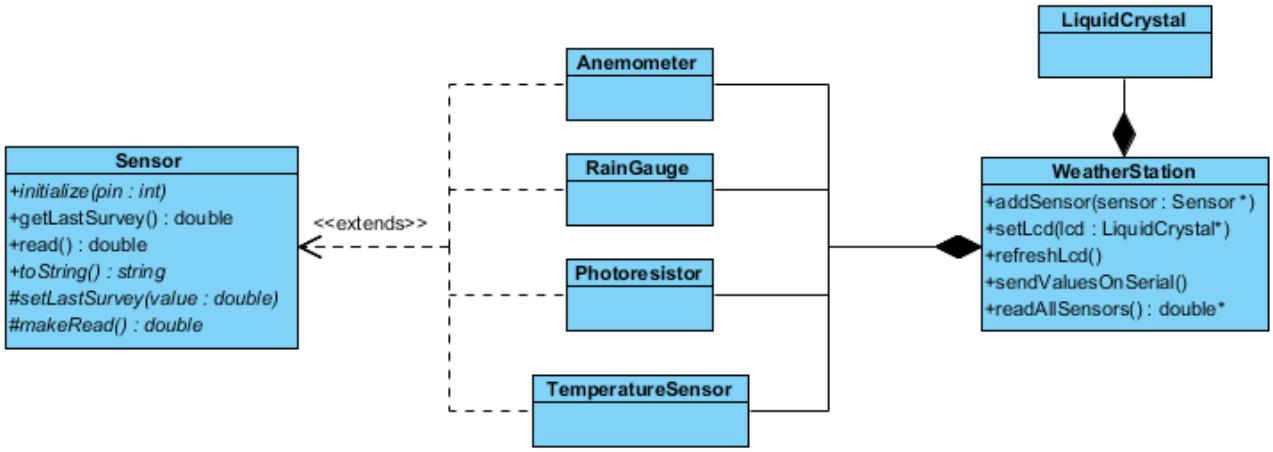
3.1 Arduino

3.1.1 Circuito

Il sistema viene alimentato tramite la presa micro-usb del Raspberry Pi con il trasformatore in dotazione standard. Arduino si autoalimenta tramite la connessione USB con il computer. E' inoltre necessario collegare il Raspberry alla rete locale tramite Ethernet o dispositivo wireless per rendere disponibili alla visualizzazione i dati raccolti.



3.1.2 Diagramma delle classi



Nello sketch di Arduino è sufficiente istanziare i 4 sensori collegati (*Anemometer*, *RainGauge*, *Phoresistor* e *TemperatureSensor*), lo schermo LCD, istanza della libreria nativa *LiquidCrystal*, e la *WeatherStation*. Quest'ultima deve essere poi configurata tramite i due appositi metodi pubblici "*addSensor*" e "*setLcd*" e impostando "*analogReference*(INTERNAL)" per la lettura dei valori analogici. Nel main loop è sufficiente quindi richiamare "*refreshLcd*" e "*sendValuesOnSerial*" quando si vuole rispettivamente visualizzare le informazioni aggiornate sullo schermo LCD e inviare i dati per il salvataggio al Raspberry. E' inoltre possibile leggere manualmente i valori tramite il metodo "*readAllSensors*".

Nello sketch utilizzato vengono effettuati l'aggiornamento dello schermo e l'invio dei dati ad intervalli fissi, rispettivamente ogni 3 secondi e ogni minuto.

3.2 Raspberry Pi

3.2.1 Organizzazione dei file

Si elencano qui di seguito le cartelle costituenti la struttura dei file del sistema sul Raspberry con una breve descrizione del contenuto delle cartelle e dei file più importanti.

Nome cartella/file	**File contenuti/descrizione file**
root/	
├── **css/**	***Stili css per le pagine web, divisi in mobile e desktop***
├── **data/**	***Database e file di ottimizzazione***
│ ├── **cache/**	***File di ottimizzazione*** (optimization-{anno}-{risol}.dat)
│ ├── arduino.db	*Database SQLite con i dati di Arduino*
│ └── sample.db	*Database SQLite con i dati d'esempio*
├── **extensions/**	***Script e classi di terzi***
├── **OOPhp/**	***Classi php scritte in object oriented***

├── **python/**	***Script in Python***
│ └── arduino_reader.py	*Lettore dei dati seriali passati da Arduino via usb*
├── DbOptimizerScript.php	*Script di ottimizzazione del database*
├── download.php	*Download dei dati in formato CSV*
├── index.php	*Pagina di welcome*
├── optimizeDb.php	*Pagina di avvio asincrono dello script di ottimizzazione*
└── ...	*Altre pagine web*

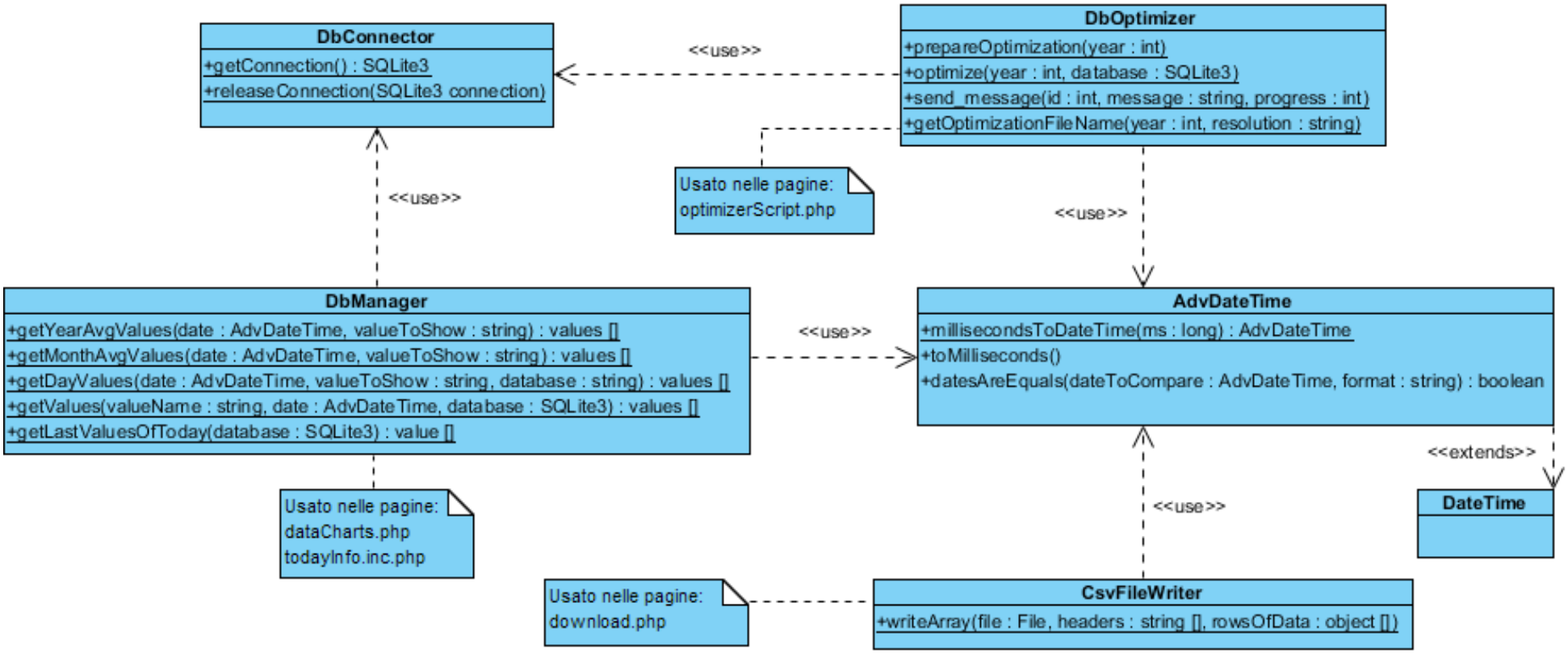
In ambiente Linux è necessario che il sistema abbia i permessi di lettura e scrittura sulla cartella *data* per il salvataggio e l'ottimizzazione dei dati. Usando *apache* è consigliato impostare l'utente e il gruppo proprietari di tali file come *www-data* (o un altro corrispondente, in base alla distribuzione scelta), utente usato dal server web nelle proprie operazioni.

3.2.2 Struttura del database

Il database presenta una struttura molto semplice, in cui ogni tupla è composta dai valori letti dai sensori di Arduino e da data e ora corrispondenti, in millisecondi. Si è scelto di dividere i dati in tabelle annuali per poter modificare la struttura delle stesse di anno in anno senza alcuna complicazione e per alleggerire il carico di elaborazione del Raspberry Pi.

Viene salvato nella cartella "*data/*" e di default con il nome "*arduino.db*".

3.2.3 Diagramma delle classi



4 Configurazione e corretto utilizzo

4.1 Parametri configurabili

Elenco qui di seguito, per ogni file, i parametri configurabili del sistema con una breve descrizione.

sketch_WeatherStation.ino

READ_CYCLES	Numero di letture da effettuare ad ogni rilevazione per temperatura e luce. Ne verrà effettuata la media, per ridurre gli errori.
LCD_INTERVAL	Intervallo di aggiornamento dell'lcd, in millisecondi.
SEND_INTERVAL	Intervallo di invio dei dati sull'interfaccia seriale, in millisecondi.
PINS_LCD, PIN_TEMPERATURE, PIN_PHOTORESISTOR, PIN_ANEMOMETER, PIN_RAINGAUGE	Pin di collegamento rispettivamente dell'LCD e dei vari sensori.
PIN_INTERRUPT_ANEMOMETER, PIN_INTERRUPT_RAINGAUGE	Pin di lettura degli interrupt per l'anemometro e il pluviometro.

arduino_reader.py

DB_PATH	Percorso del database in cui salvare i dati, comprensivo del nome del file. Es.: *../data/arduino.db.*
TIME_SLEEP	Secondi di sleep tra un ciclo e il successivo, in secondi.
SERIAL_PORT_NAME	Nome della porta seriale a cui è collegato Arduino.

DbConnector.php

DB_SAMPLE_EDINBURGH	Percorso del database contenente i dati d'esempio.
DB_ARDUINO	Percorso del database contenente i dati registrati da Arduino.

4.2 Dati d'esempio

4.2.1 Configurazione database

Per la fase di testing del sistema sono stati utilizzati dei dati d'esempio disponibili per il download sul portale della *School of GeoSciences* dell'università di Edimburgo (link).

Elenco qui di seguito la procedura per la creazione del database con la versione personalizzata di tali dati in ambiente Linux:

1. Installare sqlite3

2. Aprire un terminale posizionato nella cartella contenente il file csv con i dati d'esempio *JCMB_2014_csv_en.csv*
3. Eseguire in ordine i seguenti comandi
   a. `$ sqlite3 sample.db`
   b. `sqlite> create table data_2014 (datetime INTEGER, rainfall REAL, windspeed REAL, temperature REAL);`
   c. `sqlite> .separator “,”`
   d. `sqlite> .import JCMB_2014_csv_en.csv data_2014`
4. Spostare il database appena creato ("*sample.db*") nella cartella "*data/*" del progetto

4.2.2 Uso da interfaccia web

Per utilizzare i dati d'esempio del database di cui sopra è sufficiente aggiungere il parametro "sample" all'url desiderato.

Es.: *http://192.168.0.100/dataCharts.php -> http://192.168.0.100/dataCharts.php?sample*

Come per gli altri dati è necessario prima effettuare l'ottimizzazione per poi poter visualizzare tutti i dati correttamente.

4.3 Download dei dati

Per scaricare i dati in formato CSV da interfaccia utente si utilizza la pagina "*downloadForm.php*". Per gli applicativi è possibile richiamare direttamente la pagina "*download.php*" passando i parametri desiderati via GET, tramite i quali è possibile scaricare i dati anche in formato JSON. Si elencano qui di seguito tali parametri con una breve descrizione.

Nome	**Valori validi**	**Descrizione**
value	*all*, *temperature*, *windspeed*, *rainfall*, *brilliance*	Valori richiesti per il download. Usare "all" per scaricarli tutti.
resolution	*year*, *day*, *month*	Indica se si vogliono scaricare tutti i dati dell'anno, del mese o solo quelli del giorno selezionato.
date	*gg/mm/yyyy* (es.: 01/12/2014)	La data di riferimento.
sample	*	Se impostato il sistema usa i dati d'esempio, altrimenti quelli di Arduino.
raw	*	Se impostato il sistema invia i dati in formato JSON, altrimenti in CSV.

Esempio: *download.php?**value**=all&**resolution**=day&**date**=01%2F12%2F2014&**sample**=on&**raw***

5 Conclusioni

5.1 Problematiche

Si elencano le problematiche incontrate durante lo sviluppo del sistema.

Risolte:

- Gestione corretta ed efficiente delle **limitate risorse**:
  - *Memoria*: il sistema è stato progettato per rimanere attivo per periodi molto lunghi e anche un minimo spreco provocherebbe certamente il blocco del sistema. Per questo motivo particolare attenzione è stata posta su Arduino alla liberazione della memoria allocata dinamicamente e al possibile overflow delle variabili utilizzate, limitando tale rischio per quanto possibile.
  - *Computazione*: i calcoli più lunghi e complessi sono stati messi tutti a carico del Raspberry ma tale risorsa rimane comunque limitata. Per questo i dati vengono elaborati e salvati prima di poter essere visualizzati nei grafici in pagina web, diminuendo i tempi di caricamento della stessa e il carico del processore per ogni visualizzazione.

In sospeso:

- Libreria per il disegno dei **grafici**:

  La libreria utilizzata per il disegno dei grafici (Google Charts), a causa dei termini di servizio, non è legalmente utilizzabile offline. Per il momento sono stati scaricati i file necessari al suo funzionamento, ma se fosse necessario utilizzare il sistema in un ambiente senza connessione bisognerà utilizzare una diversa libreria.

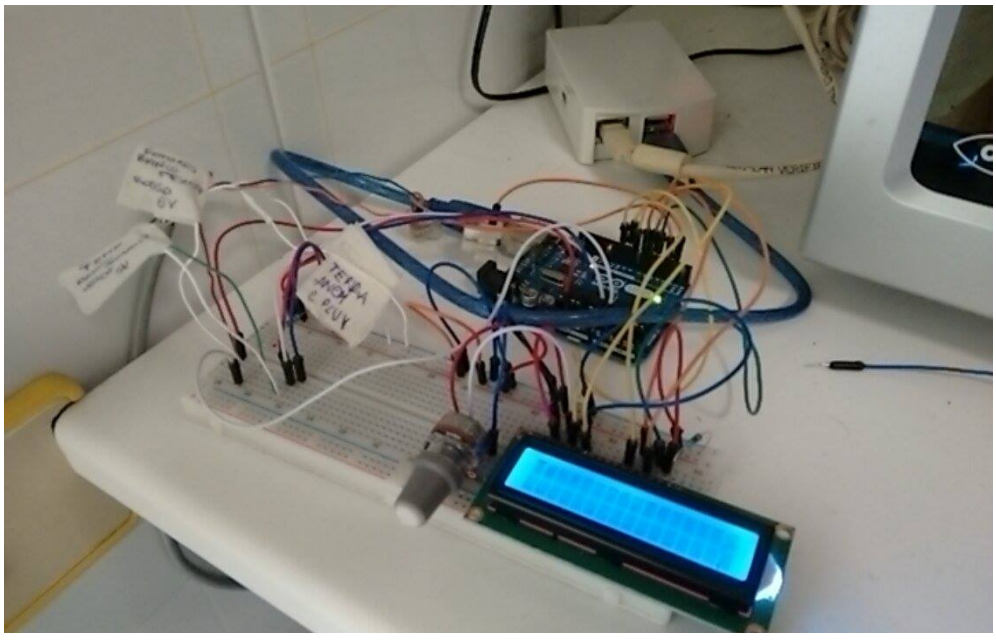
5.2 Testing

In primis è stata testata l'interfaccia web con l'utilizzo dei dati d'esempio. Il sistema ha risposto bene nonostante la grande quantità di dati utilizzati (in media una rilevazione al minuto), l'ottimizzazione ha permesso di ridurre notevolmente i tempi di caricamento per la visualizzazione dei dati in grafici.

Dopodiché è stato aggiunto il sistema di rilevazione per un secondo test durato circa due settimane, inizialmente con tutti i sensori posizionati internamente all'abitazione per valutarne la capacità di rimanere attivi nei lunghi periodi e successivamente posizionandoli all'esterno, sufficientemente in alto da poter effettuare rilevazioni realistiche. Anche qui il sistema si è rivelato ben programmato e in grado di svolgere i propri compiti con buoni risultati. Le uniche problematiche rilevate e degne di nota sono state: un aumento della temperatura media rilevata, dovuta all'utilizzo di cavi e strumenti non professionali e alle giunture non saldate, e il blocco della fotoresistenza sul valore massimo, dovuto alla permeabilità del proprio contenitore e alla forte pioggia che ha chiuso il contatto escludendo la fotoresistenza.

Per il testing è stato aggiunto lo script Python "*arduino_reader.py*" a *crontab*, un processo eseguito in background che permette l'esecuzione di script in determinati momenti temporali. Tale script è stato impostato per l'avvio ad ogni riavvio del Raspberry Pi per assicurarsi la sua esecuzione anche in caso di riavvio del sistema operativo. In questo modo bisogna porre particolare attenzione solo al caso in cui il programma di lettura registri delle eccezioni dovute ad errori nella sua esecuzione. Nel testing è stato impostato un file di log in cui viene sempre stampato il PID del processo lanciato e gli stack trace delle eventuali eccezioni.


Vento: 5.57 kmh
Piogg: 0.00 mm



5.3 Installazione ottimale

Per un'installazione ottimale del sistema è necessario installare i sensori in modo da ottenere le migliori rilevazioni possibili da ognuno di essi.

Le maggiori attenzioni vanno riservate all'anemometro, che dev'essere posizionato molto in alto, almeno al di sopra della casa su cui è installato, e al sensore di temperatura, che dev'essere posizionato in un luogo mai ombreggiato e protetto da un contenitore di materiale isolante ed arieggiato. In questo modo il sensore rimane nelle stesse condizioni durante tutto l'arco della giornata, non riceve la luce diretta e le rilevazioni non vengono condizionate dalla prolungata esposizione.

Infine è necessario proteggere la circuiteria con contenitori stagni isolanti per evitare il contatto con la pioggia, l'umidità e i piccoli animali.

5.4 Possibili utilizzi ed integrazioni

Il sistema è stato progettato per l'utilizzo all'interno di un impianto smart-home. Tramite le rilevazioni effettuate è possibile automatizzare un elevato numero di operazioni casalinghe come ad esempio un sistema di irrigazione programmato in base alla quantità di pioggia caduta, l'accensione delle luci esterne al calare della notte, la chiusura di porte e finestre in caso di forte vento o pioggia o in base alla temperatura esterna.