Un sistema per il monitoraggio di impianti fotovoltaici

Progetto e implementazione

Loris Fichera

Relatore: Prof. Corrado Santoro

Università degli Studi di Catania
Corso di Laurea Specialistica in Ingegneria Informatica

20 Luglio 2011

Monitoraggi *immaturi*

Il numero di impianti fotovoltaici *grid-connected*, in Italia, è in costante aumento

- potenza installata raddoppia ogni anno
- oltre 4 GW al 31/12/2010

Monitorare gli impianti fotovoltaici diventa importante per

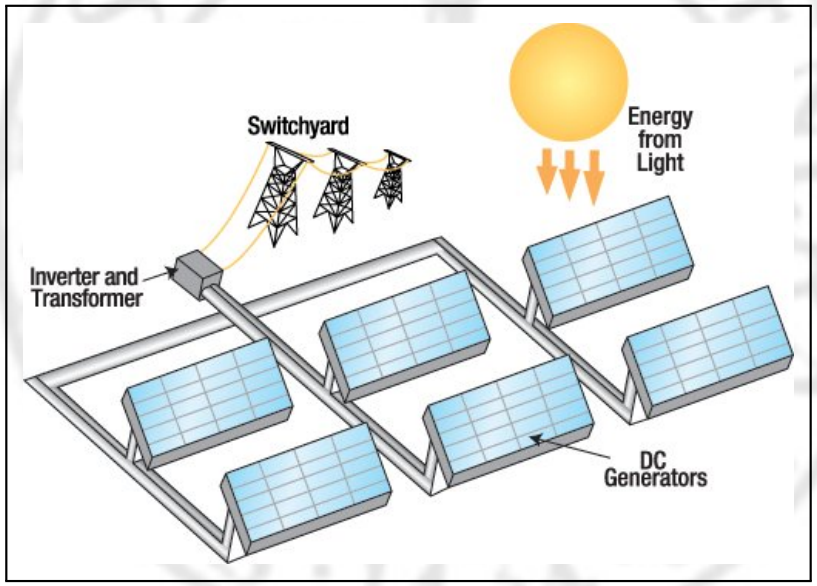
- i soggetti responsabili
- gli installatori/manutentori

"Gran parte delle soluzioni oggi in commercio mostrano caratteri di *immaturità*" (C. Podewils 2010)

Obiettivi

Un sistema di monitoraggio effettua la raccolta e l'integrazione dei dati rilevanti di un impianto al fine di determinarne:

- lo stato operativo
- l'efficienza globale
- la produzione energetica



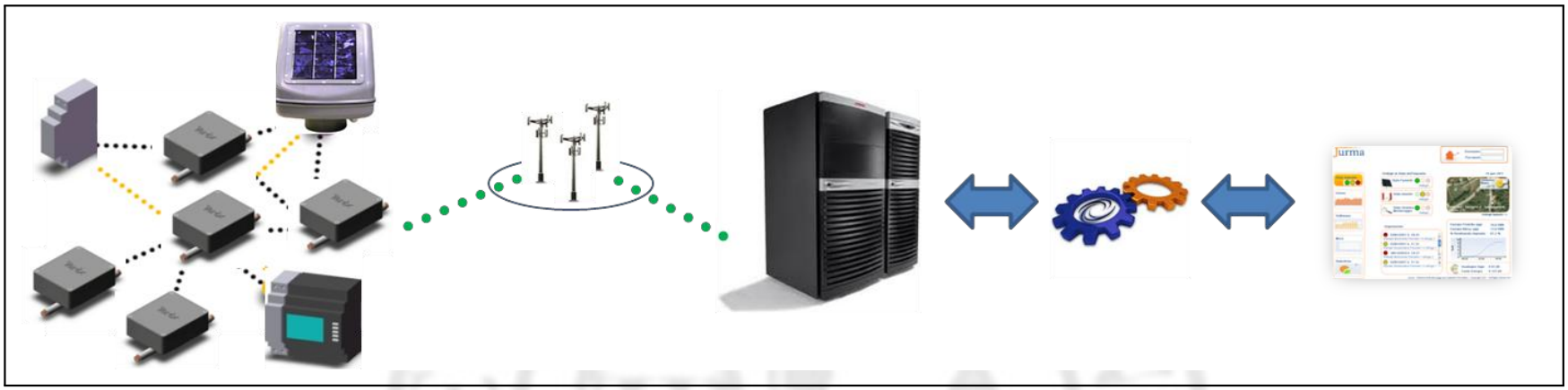
Quali sono i dati rilevanti?

Classi di utenti (Kolodenny 2008)

Kolodenny *et al.*, identificano le seguenti classi di utenti:

- ricercatore
  - quanti più dati possibile
  - alto livello di dettaglio
- proprietario
  - energia prodotta
  - ritorno economico
- manutentore
  - comportamento di ogni componente (modulo, inverter...)
  - dati ambientali

Panoramica del sistema



- infrastruttura di campo
- trasmissione dati via gprs (o sms)
- datacenter
- accesso ai dati via web services
- portale web

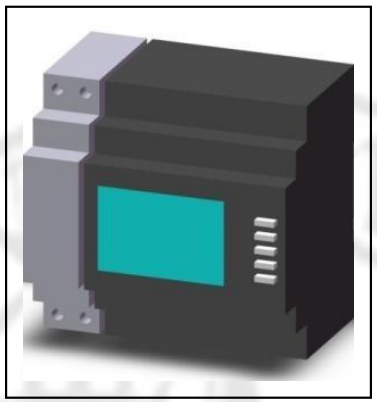
Infrastruttura di campo

- Wireless Sensor Network (WSN) ZigBee-based
  - bassi costi
  - no cablaggi aggiuntivi
  - rete autoconfigurante
- previsti tre tipi di nodi
  - gateway
  - power/inverter transponder
  - string transponder

Il Gateway

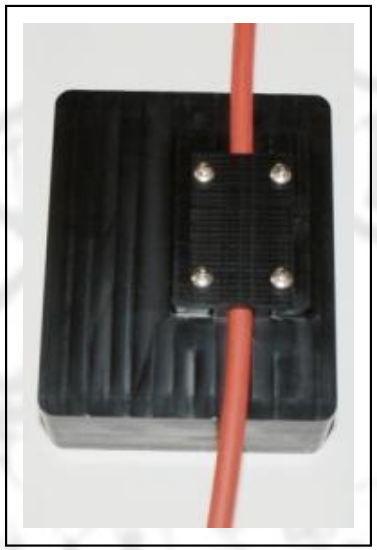
- agisce da concentratore dati
- modem gsm per la trasmissione dei dati
  - via sms
  - via FTP *over* gprs
  - transmission interval configurabile
- elemento fotovoltaico per la ricarica della batteria

Il Power/Inverter Transponder



- analizzatore di rete + transponder ZigBee
- misura delle grandezze elettriche (corrente, potenza, power factor...)
  - a monte del contatore bidirezionale
  - a valle di ogni inverter
  - sampling interval configurabile

Lo String Transponder



- sensore di corrente (a *effetto Hall*) + transponder ZigBee
- misura della corrente di stringa
  - sampling interval configurabile

Il Datacenter

Si occupa di

- gestire i flussi informativi generati dai gateway
- elaborare e memorizzare i dati di campo
- generare report e allarmi
- fornire accesso ai dati

- è interamente basato su Erlang/OTP
- è costituito da un insieme di Erlang applications

L'applicazione database

Possiede un unico processo worker, il quale funge da wrapper per il database MySQL in cui vengono memorizzati

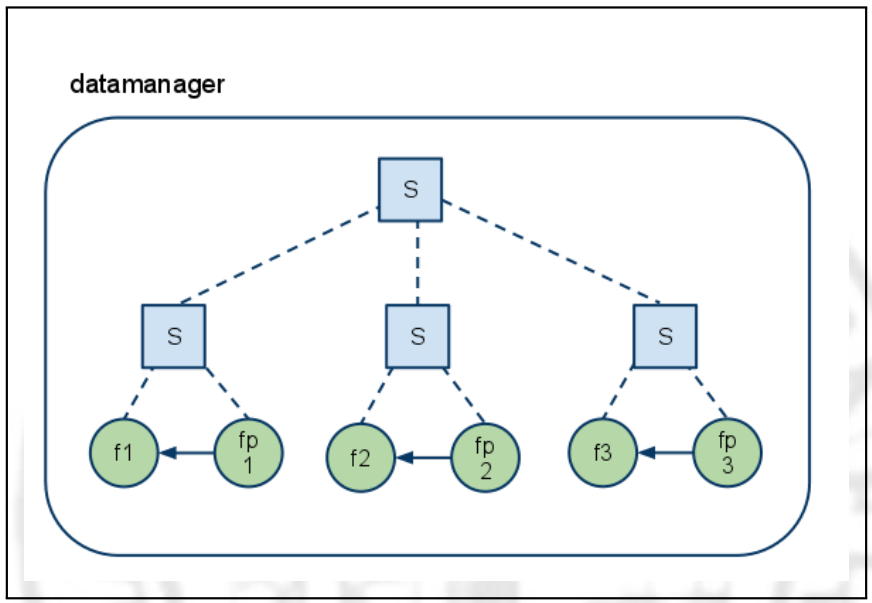
- configurazioni degli impianti
- dati di monitoraggio

Code snippet

```
data_storage:add_new_plant_by_structure (
      _UniqueID = "MyPlant",
      _Location = "Nowhere",
      ...
      _Inverters = ["YURAKU", "I-3000",
                    "Inverter-00", "0000-0000-0000",
                    [%% strings here...
                     ] ]).
```

L'applicazione `datamanager`

Gestisce i flussi informativi generati da ogni impianto



- un *process pool* per ogni impianto
- due worker per *pool*
  - un `filter`
  - un processo endpoint (`file_poller`, `sms_manager`)
- politiche di riavvio dei supervisor

Il `file_poller`

- decodifica i dati ricevuti via FTP
- genera report riguardo
  - dati corrotti
  - dati mancanti
- invia i dati ricevuti al `filter`

Decodifica dei dati

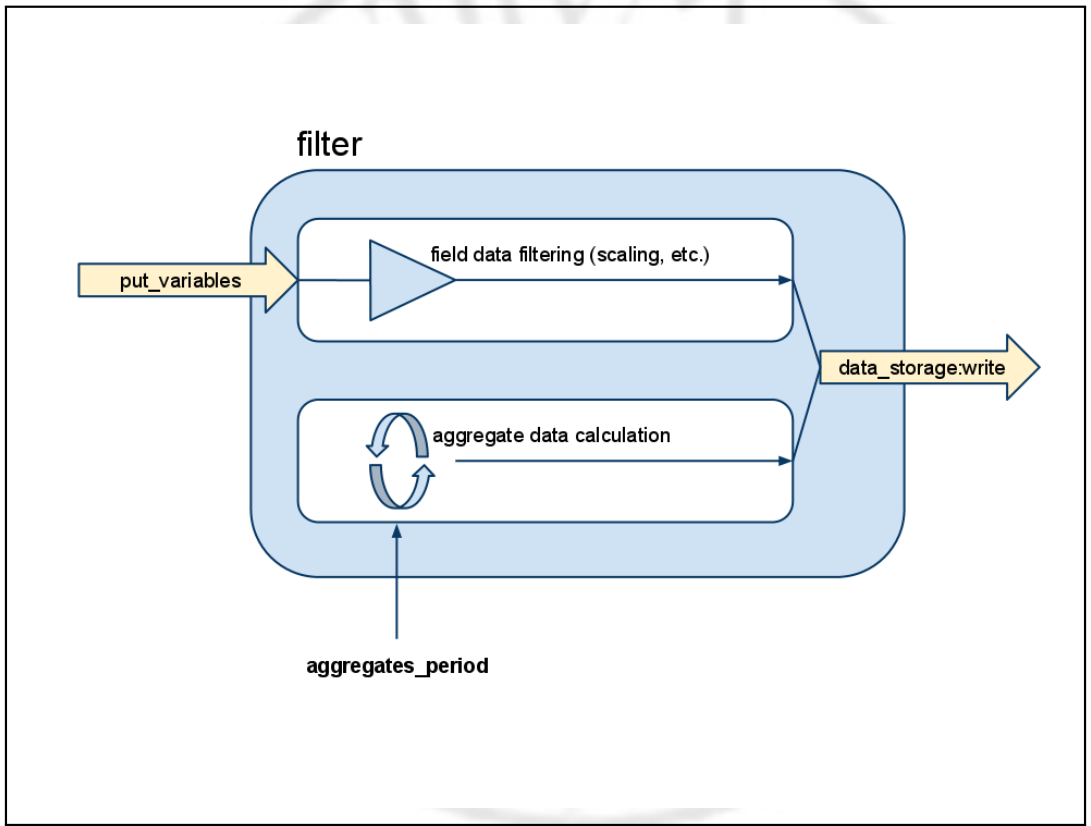
```
try
 {ok, FileContent} =
   file:read_file (lists:concat ([Path, File])),
   ftp_protocol:decode (FileContent)

catch error : {Reason, D} ->
    ...
  nil
end,
```

Il `filter/1`

Due attività concorrenti:

- filtraggio dei dati ricevuti (scaling, ecc.)
- stima di altri valori



Il `filter/2`

Calcolo della potenza totale per Power Transponder

```
{ok, {_, Vars}} =
  data_storage:get_trend_by_device (DeviceID, F, T,
                                    ["active-power-l1",
                                     "active-power-l2",
                                     "active-power-l3"]),

PropList = measure_data_to_proplist (Vars),
P1 = proplists:get_value ("active-power-l1", PropList),
P2 = proplists:get_value ("active-power-l2", PropList),
P3 = proplists:get_value ("active-power-l3", PropList),

TotalActivePower = P1 + P2 + P3,
```

Web Services

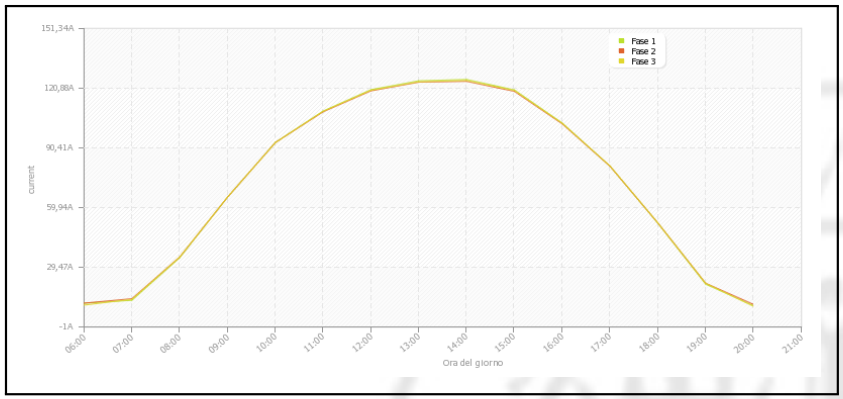
- basati su Yaws
- RESTful

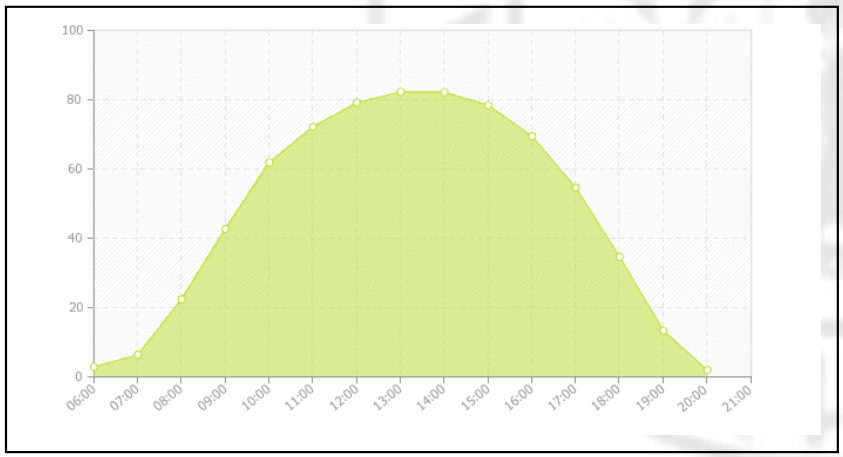
Servizio get_plants

```
<erl>
out(A) ->
 {ok, PlantList} = data_storage:get_plants(),
  Reply =
    xml_translator:make_plant_list_response (PlantList),
  html, Reply.
</erl>

<plant_list_response>
  <plant id="1" unique_id="plant-1" customer_id="0" ... />
</plant_list_response>
```

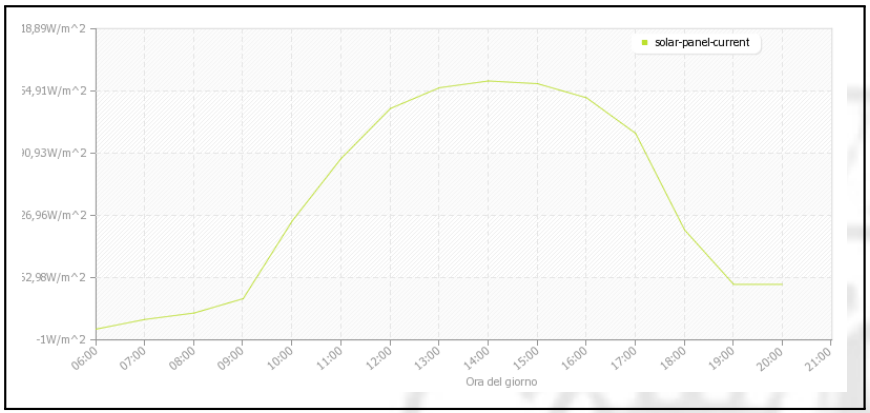
Potenza prodotta vs. Corrente immessa in rete



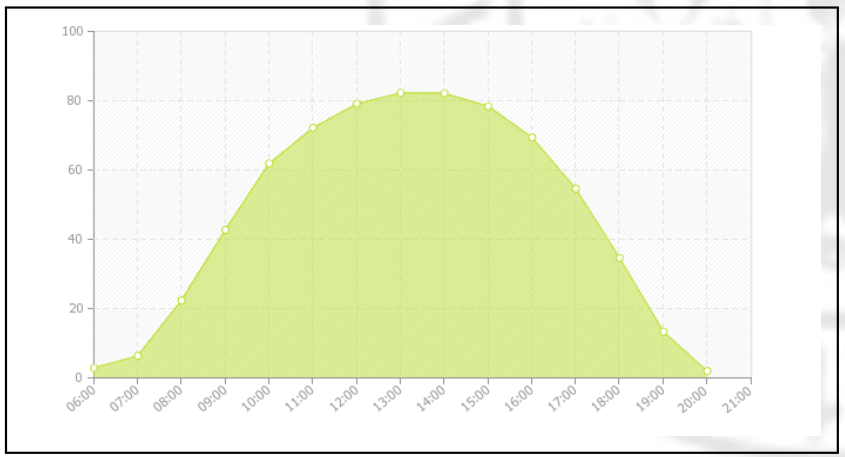


blah
blah

Potenza prodotta vs. Radiazione



blah
blah



Sviluppi futuri

- sicurezza, blah
- sistema esperto per la diagnosi, blah