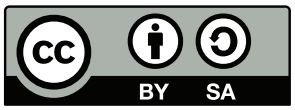
Studio del calcolo della potenza di un carico interrompibile in ambiente domestico

Francesco Bartolini

4 aprile 2016



Introduzione

Lo studio presentato in questo articolo fa parte di un progetto più grande che riguarda la creazione di una simulazione di un sistema domotico basato su hardware a basso costo come Raspberry Pi, Arduino e contatori ad impulsi, al fine di poter gestire ed ottimizzare il consumo energetico domestico coadiuvato da pannelli fotovoltaici e storaging energetico. Lo studio propone di elaborare un algoritmo, il più possibile accurato e versatile, per la stima della potenza consumata da un carico interrompibile all'interno di un'abitazione domestica.
Più in dettaglio, il sistema creato prevede l'uso di Rasperry Pi B+ come "cervello" del sistema e di due device Arduino UNO per interfacciarsi, rispettivamente, con il carico interrompibile (per spegnerlo ed accenderlo) e con il contatore ad impulsi che, collegato al carico stesso, permette di misurare la potenza consumata; il tutto è interconnesso in una rete ethernet.
Dato l'utilizzo di un contatore ad impulsi per calcolare la potenza istantanea consumata dal carico, si è dovuto approfondire lo studio del calcolo di tale potenza a causa delle limitazioni intrinseche che ha un contatore ad impulsi per tale misurazione. Nella fattispecie il contatore scelto emette 1000 impulsi ogni kWh misurato e di conseguenza, attraverso un calcolo, si può misurare la potenza consumata istantaneamente. Il fatto che un contatore ad impulsi non estragga un valore reale della potenza ma, come dice il nome stesso, conti gli impulsi emessi in base al flusso di energia elettrica transitata in esso, ha richiesto un cambiamento riguardo l'approccio alla "misurazione istantanea" ed all'idea di "stima finale della potenza".
Il contatore, come già accennato, emette un tot di impulsi per kWh, nel nostro caso 1000 Imp/kWh ovvero un 1 impulso per Wh. Quindi se in un'ora

ci fosse una variazione di impulsi pari ad uno, vorrebbe dire che è stata consumata una quantità di energia pari ad 1 Wh. Il nostro scopo è però quello di utilizzare il contatore per fornire una misurazione di potenza (lavoro al secondo) quasi istantanea, e non una misurazione media della potenza in un'ora. Perciò si è pensato di misurare la differenza degli impulsi contati dal contatore (ed interposti dalla board Arduino) in un minuto e moltiplicare questa cifra per 60, che rappresenta la potenza media richiesta in tale minuto, calcolata in W. Per esempio, se avvessimo al minuto 1 un valore di impulsi pari a 20, e al minuto 2 un valore pari a 22, avremmo:

```
(22 - 20) * 60 = 120
```

Il valore ottenuto, 120, risulta essere la potenza in Watt calcolata in quest'ultimo minuto. Ovviamente il limite del calcolo della potenza consiste proprio in questo: la misurazione viene fatta ogni minuto e viene calcolata confortando il valore attuale con quello precedentemente ottenuto. Attenzione però: si potrebbe calcolare anche in ogni secondo (e quindi moltiplicando per 3600), ciò non toglie che ci si dovrebbe rapportare con il concetto di differenza di impulsi e soprattutto con il fatto che per consumi di Watt piccoli non si avrebbero differenze di impulsi nell'arco di un buon numero di secondi. La scelta di calcolare la potenza ogni minuto infatti è maturata durante la progettazione. Si è notato che per carichi alti (ad esempio 1000 W) la misurazione per secondo poteva essere una buona scelta. Ma per carichi bassi (ad esempio 100 W) sarebbe stato uno spreco di risorse computazionali e di banda inutile, proprio dovuto al fatto che per "misurare" la potenza di un carico piccolo (o meglio, far variare il numero degli impulsi del contatore, collegato ad un carico piccolo) si necessita di un maggiore numero di secondi.

Alla luce di ciò, il problema da risolvere consiste nel creare un buon algoritmo di stima che operi ottimamente anche su impostazioni precostituite che potrebbero limitarne l'accuratezza dei risultati e che, allo stesso tempo, sia anche versatile dal punto di vista della potenza reale dissipata dal carico interrompibile in questione.
Ogni minuto viene preso il valore degli impulsi contati e viene misurata la potenza in quel minuto.
Si è perciò deciso di simulare una circostanza verosimile alla realtà, creando uno scenario di utilizzo del carico in questione da parte dell'utente. Lo scenario consiste nella rappresentazione, nell'arco di 120 minuti, di una serie di accensioni ed spegnimenti del carico fini alla simulazione di una situazione quotidiana, con lo scopo di verificare teoricamente, ed a priori, la correttezza degli algoritmi studiati, confrontando i risultati con la potenza reale dissipata dal carico. In sostanza, sono testati degli algoritmi di stima della potenza in un ambiente e scenario verosimile alla realtà e vengono poi confrontati gli scostamenti e gli errori dei risultati, prodotti da ogni algoritmo, dalla poten-

za reale prestabilita. Viene inoltre studiata la differenza che ogni algoritmo produce per potenze diverse in un range che può essere presente all'interno di un'abitazione domestica. Infatti la quantità di energia elettrica che un provider può concedere ad un'ambiente domestico è pari a 3300 Wh (ed al superamento di tale soglia per oltre 3 minuti viene dismessa l'erogazione di energia per ragioni di sicurezza e di contratto stabilito con il provider stesso). Quindi i vari algoritmi sono stati testati per ogni carico di potenza che varia da 1 a 3300 Watt.
Per creare lo scenario e gli algoritmi è stato utilizzato il linguaggio Python (v. 3) e librerie esterne molto conosciute in ambiente scentifico come NumPy.

Creazione Scenario

Come detto lo scenario è rappresentato da 120 minuti di accensione e spegnimento del carico. Quindi, a livello astratto, tale situazione può essere rappresentata come un array nel quale ogni cella raffigura la situazione istantanea del minuto in cui viene calcolata la stima della potenza. Perciò ogni indice dell'array rappresenta un minuto da 0 a 120 minuti: l'array, dunque, è di lunghezza pari a 120, ed ogni cella può contenere due valori, 0 per la rappresentazione dello spegnimento del carico, e 1 per l'accensione.
La struttura necessaria alla raffigurazione del problema necessità però di altre informazioni, come la potenza reale, il valore della potenza calcolato dall'algoritmo, lo scostamento dalla potenza reale, etc..quindi un semplice array non può essere sufficiente. Si è deciso di utilizzare un array di array con lo scopo di rappresentare una sorta di tabella con ogni riga formata da un array, ed ogni array, attraverso le celle, che forma le colonne dove vengono rappresentati la situazione di accensione/spegnimento del carico, il valore degli impulsi calcolati in base alla potenza reale (quindi con virgola mobile), il valore degli impulsi virtualmente contati dal contatore ad impulsi (quindi il valore precedente senza virgola mobile), e poi i vari algoritmi di stima e le differenze con la potenza reale.
Il vantaggio di utilizzare una struttura del genere risiede nel fatto di poter aggiungere o rimuovere colonne a piacimento. Quindi, potenzialmente, possono essere aggiunte `n` colonne per `n` algoritmi di stima ed i propri scostamenti dal valore reale, aggiungere vari carichi e vari valori degli impulsi per contatori diversi, etc.
Il codice che crea il tutto (sia lo scenario che le varie stime) è all'interno di un unico file, `benchmark.py`. Analizziamo la prima parte.
I moduli importati sono `json`, per l'utilizzo di strutture e file `JSON`, `sys` per leggere e scrivere su file, e `numpy`. Il modulo `numpy` [1] è un estensione del linguaggio `Python` che facilita la creazione di array multidimensionali e matrici, aggiungendo la possibilità di utilizzare un gran numero di funzioni

matematiche ad essi associate, per la manipolazione dei dati. `NumPy` è un progetto opensource e community driven, ed è largamente utilizzato in ambito scientifico per i più svariati calcoli.

Il codice mostra che viene utilizzato un file `params.json`. Tale file contiene una struttura `JSON` dove vengono rappresentati in coppie chiave/valore il minuto e la situazione di accensione (1) o spegnimento (0) del carico. Da questo file, quindi, viene creato lo scenario.
La fase di pre-popolamento della "matrice" utilizza i dati appena caricati dal file per sapere quante righe (e quindi quanti minuti) devono essere create, e per formare delle liste Python che serviranno al popolamento della matrice stessa, precisamente al popolamento della colonna che riguarda lo stato di accensione/spegnimento del carico.
Con il `for` associato alla funzione `enumerate()` si apre il ciclo che permette di creare i 120 minuti di scenario per ogni potenza che varia da 1 a 3300 Watt. Vengono perciò creati 120 array per 3300 valori di potenza diversi.
Gli array stessi vengono creati con la funzione `ndarray` di `numpy`. Le righe saranno quante sono le chiavi del file `params.json` e le colonne sono 12, in quanto servono colonne per:

1. Numero del minuto
2. Valore Accensione o Spegnimento (1 o 0)
3. Numero Impulsi al secondo
4. Numero Impulsi al minuto
5. Stima Algoritmo 1
6. Errore Algoritmo 1
7. Stima Algoritmo 2
8. Errore Algoritmo 2
9. Stima Algoritmo 3
10. Errore Algoritmo 3
11. Stima Algoritmo 4
12. Errore Algoritmo 4

Dal codice precedente popoleremo la matrice solo fino alla quarta colonna; i dati per popolare le rimanenti colonne sono creati dagli algoritmi testati che, come è stato già accennato, sono quattro.

Studio e Testing degli algoritmi

Primo algoritmo di stima

Il primo algoritmo di stima è molto semplice. Per ogni minuto (quindi per ogni array con lo stesso valore di potenza reale) calcola la differenza tra gli impulsi al minuto del valore corrente con quello del precedente ed il risultato lo moltiplica per 60.
Poi riempe la quinta colonna con la differenza tra il valore di stima calcolato e la potenza reale.
Tale soluzione non può essere considerata soddisfacente, il valore stimato si discosta troppo dal valore reale ed inoltre non rimane neanche costante ma oscilla nel tempo.

Secondo algoritmo di stima

Il secondo algoritmo utlizza invece la differenza tra il valore degli impulsi al minuto corrente ed il valore degli impulsi calcolato 5 minuti prima, per poi dividere il risultato di tale differenza, moltiplicata per 60, per il numero di elementi presenti tra i due valori (compresi), quindi 6.
Seppur la stima calcolata risulta essere molto più accurata dell'algoritmo precedente, vi è un problema di velocità nel raggiungimento del valore calcolato che porterebbe quindi ad un peggioramento della reattività nella segnalazione di un carico troppo alto. Quindi anche in questo caso l'algoritmo deve essere scartato.

Terzo algoritmo di stima

Il terzo tentativo ha portato alla creazione di un algoritmo che applica il calcolo della potenza come nel primo algoritmo e aggiunge questo valore in un buffer. Poi somma tutti i valori presenti all'interno del buffer e li divide per il loro numero.
Valgono solo i valori positivi, quindi si applica una sorta di media tra tutti gli ultimi valori di potenza calcolati (con il primo algoritmo) fino all'ultimo valore precedente (a ritroso quindi) maggiore di zero. Il valore ottenuto è la stima della potenza per quel minuto.
Questo algoritmo raggiunge un buon compromesso tra accuratezza e reattività nel calcolo del valore. Inoltre è anche abbastanza costante nel mantenere il valore.

Quarto algoritmo di stima

L'algoritmo n° 4 si comporta come il terzo, ma controlla che ci sia una variazione d'impulsi. Se questa non è presente, allora potrebbe non voler dire che il carico sia spento ma che la potenza sia troppo bassa da poter aumentare gli impulsi. Quindi la stima 4 è come se "rallentasse" la stima 3. L'algoritmo 4 risulta peggiorare la stima dell'algoritmo 3, in quanto si accorge in ritardo sia dell'accensione del carico che del suo spegnimento.

Scelta Finale

L'algoritmo migliore dal punto di vista dell'accuratezza e del rapidità nella risposta già dai risultati precedenti risulta essere il terzo. Ma il codice sottostante certifica ancora di più la correttezza della scelta.
Infatti per ogni valore di potenza reale compresa tra 1 e 3300 Watt, calcola le stime dei quattro algoritmi nello scenario prestabilito (120 minuti, con un certo numero di minuti di accensione e spegnimento). Per ogni stima, ricava l'errore medio in 120 minuti. Con questo valore calcola l'errore, confrontandolo con il valore reale, in percentuale, così da avere l'errore medio nei 120 minuti per algoritmo per ogni valore di potenza (compresa tra 1 e 3300 Watt). Questo valore in percentuale viene moltiplicato da un valore riconducibile ad un "peso" che può avere il valore stesso di potenza reale. Perciò con valori minori di 50 Watt, il peso è pari a 1; da 50 Watt in poi ed ogni 100 Watt di valore, il peso aumenta di uno. Vengono quindi sommati per ogni stima i propri errori medi (nei 120 minuti) in percentuale e pesati con l'importanza della potenza reale. Infine avremo quindi 4 valori che verranno divisi per la somma dei valori "peso" precedentemente assegnate alle potenze.
Il risultato è:

```
The final values are:
0.4414203053462314 2.0999341029340988 0.12084567010492926
0.9166929791744616
...and the winner is
 -> STIMA 3 <-
```

Il tutto viene salvato in un file `results.txt`. Quindi è stato dimostrato che l'algoritmo migliore risulta essere il **terzo**.

Riferimenti bibliografici

[1] Travis Oliphant. Numpy. `http://www.numpy.org`.