ROMA TRE
UNIVERSITÀ DEGLI STUDI

Scuola Dottorale EDEMOM

European Doctorate in Electronic Materials, Optoelectronics and Microsystems

XXVIII Ciclo

Implementazione di una Infrastruttura completa per il Monitoraggio della Power Quality

Dottorando Stefano Di Pasquale

Tutor Prof. Maurizio Caciotta

Aprile 2016

Indice

Lista delle Figure

Introduzione

L’estrema difficoltà che si ha ad immagazzinare grandi quantità di energia elettrica ha reso necessario lo sviluppo di un’efficace rete di distribuzione in grado di permetterne l’utilizzo, da parte di un utente, praticamente nell’istante stesso in cui questa viene prodotta ed anche se questo utente si trova a centinaia di chilometri di distanza dalla centrale elettrica di produzione.

Ecco quindi che con la nascita del sistema elettrico ha avuto origine anche lo studio dei problemi legati alla distribuzione dell'energia elettrica, infatti, affinché l'energia sia usufruibile non è sufficiente generarla e trasportarla all'utilizzatore, ma bisogna garantire che arrivi a quest’ultimo con una ben determinata "forma", ovvero con ben determinate caratteristiche.

L’energia elettrica che viene distribuita in Italia ed in Europa, idealmente, presenta le caratteristiche principali di avere una forma d’onda sinusoidale con frequenza di 50 Hz ed una tensione efficace di 230 V. Andando però, ad analizzare le forme d’onda che arrivano ad un utilizzatore si trova, spesso, qualcosa di differente e che cambia nel tempo e da punto a punto di osservazione.

Questo discostamento dell’energia elettrica dalla sua idealità può avere diverse cause, ognuna che incide diversamente a seconda del caso:

- la complessità ed irregolarità della rete di distribuzione, costituita da svariati chilometri di conduttori e da numerosi trasformatori che si incontrano lungo il percorso;
- la grandissima varietà di carichi elettrici connessi alla rete, ognuno con le proprie caratteristiche di assorbimento ed i propri intervalli di funzionamento;
- la disomogeneità nella distribuzione sul territorio di questi carichi;
- la crescente presenza di Generazione Distribuita, che presenta spesso i suoi convertitori elettronici per adattare la forma dell’energia elettrica generata a quella dell’energia distribuita sulla rete a cui è connessa.

Perciò questo "discostamento dall'idealità" non dipende solo da come viene generata l'energia elettrica e dal suo sistema di trasporto, ma dipende anche dagli utenti che utilizzano l'energia stessa e che sono in grado con i loro carichi, spesso inconsapevolmente, di modificarne le caratteristiche sia per loro che per gli altri utenti della rete.

Quindi la "qualità della fornitura di energia elettrica" che arriva all'utente finale, intesa, per ora, come il discostamento dalle forme d'onda ideali, può essere ben più scarsa di quella dichiarata dal distributore a causa di tutto ciò che ruota attorno alla rete di distribuzione, utenti inclusi.

Oggigiorno gli utilizzatori dell'energia elettrica sono sempre più sensibili a questo argomento, in quanto una "bassa qualità" dell'alimentazione è in grado di compromettere il corretto funzionamento delle apparecchiature ad essa collegate e quindi, ad esempio in ambito industriale, può condizionare fortemente il normale processo produttivo e causare intollerabili disservizi. Questo si traduce, molto spesso, in perdite economiche anche di notevole entità.

In questo contesto, in cui si inserisce il settore della Power Quality, il monitoraggio della qualità dell'energia acquista un ruolo fondamentale per ottenere informazioni sullo stato di una sottorete elettrica, utili non solo per individuare e quantificare i disturbi ma anche per cercare di determinare delle responsabilità. Inoltre, in alcune situazioni, avendo un'ottima conoscenza della rete, potrebbe anche essere possibile prevederne il comportamento ed agire in tempo per prevenire delle anomalie o il peggioramento della situazione.

Perciò l'obiettivo di questo lavoro è stato quello di ***sviluppare un'infrastruttura completa per il monitoraggio della Power Quality***, da poter distribuire geograficamente su larga scala, in modo da aumentare notevolmente la conoscenza dello stato della rete elettrica. Quindi, realizzare una struttura con dei punti di monitoraggio che abbiano bassi costi, che siano compatti, affidabili e capaci di fornire dei dati riferibili temporalmente, in modo da poter essere confrontati rispetto ai vari siti. Inoltre, questa struttura deve essere in grado di visualizzare agevolmente tutti i dati di Power Quality acquisiti e di gestire facilmente numerosi punti di osservazione che potrebbero essere dislocati ovunque sul territorio.

Si parla di infrastruttura "*Completa*" perché è stata sviluppata sia la parte hardware, per acquisire i segnali fisici dalla rete, sia il software per elaborarli e visualizzarli, sia,

soprattutto, un meccanismo per la gestione ed il controllo, completamente da remoto, dei vari siti, cosa non affatto banale e spesso non presente nei sistemi di monitoraggio.

Questo lavoro è stato suddiviso in 3 sezioni principali.

Nella prima verrà data una breve descrizione della Power Quality, menzionando quali possono essere gli effetti di una cattiva alimentazione, classificando i maggiori disturbi e fornendo alcune informazioni sulle principali normative in vigore nel settore della PQ e della sua strumentazione. Inoltre, si parlerà dei sistemi di monitoraggio e verrà mostrato il vecchio sistema realizzato ed utilizzato dal laboratorio di Misure Elettriche ed Elettroniche.

Nella seconda parte verranno dapprima elencati gli obiettivi da raggiungere e poi verrà descritta, in dettaglio, la "nuova" infrastruttura realizzata. Questa descrizione è stata articolata esponendo separatamente i componenti dei 3 aspetti principali dell'infrastruttura. Prima verrà descritta la parte hardware, il cui elemento principale è il BeagleBone Black che ha permesso di ridurre notevolmente costi e dimensioni, aumentando l'affidabilità, poi il software ed in ultimo la parte dedicata al sistema di gestione e controllo, entrambi basati sul web.

Infine, nell'ultima parte, sono state tratte delle conclusioni, mostrando gli obiettivi raggiunti e riassumendo le caratteristiche principali della nuova strumentazione.

Inoltre, in Appendice, è stato inserito il codice (commentato) di alcuni programmi e script sviluppati e descritti nel testo.

La Power Quality

Idealmente, la migliore tensione di alimentazione elettrica dovrebbe avere una forma d'onda sinusoidale con frequenza e ampiezza costanti, tuttavia, a causa della grande diversità dei carichi elettrici connessi, a causa dell'impedenza non nulla della rete elettrica ed a causa di altri fenomeni come transitori e interruzioni, la realtà è spesso molto differente [Schipman].

Infatti, secondo i testi di elettrotecnica, si ha che l'energia elettrica che viene distribuita in Italia ed in Europa sotto forma di tensione alternata, prendendo in considerazione un sistema trifase, ha un andamento del tipo mostrato in Figura 2.1.

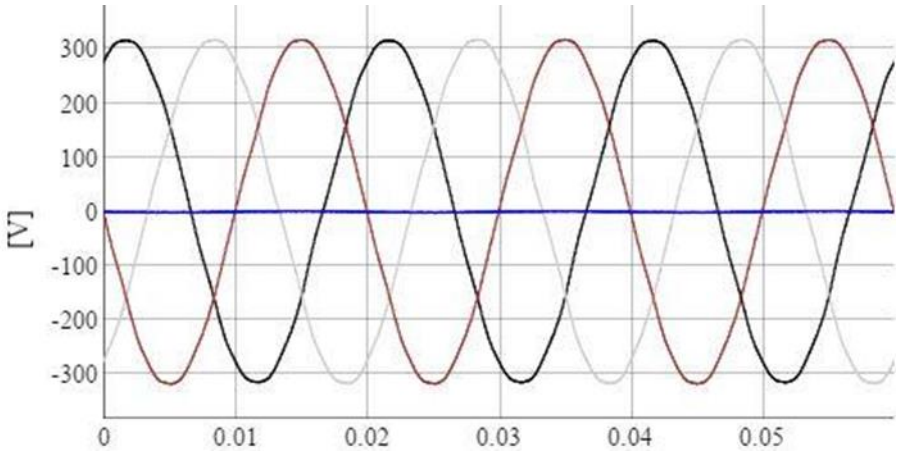


Figura 2.1 - Andamento delle tensioni in un sistema trifase ideale.

Le caratteristiche principali di questo andamento sono quelle di avere una forma d'onda costituita da solo 3 sinusoidi con frequenza di 50 Hz, una tensione efficace di 230 V (per fase) ed il fatto che le sinusoidi delle tre fasi R-S-T, che formano il sistema trifase, sono sfasate tra loro di 120°.

Quindi, per le tensioni, si ha una situazione del tipo:

$$v_R(t) = \sqrt{2}V \sin(2\pi ft + \varphi_R)$$
$$v_S(t) = \sqrt{2}V \sin(2\pi ft + \varphi_S)$$
$$v_T(t) = \sqrt{2}V \sin(2\pi ft + \varphi_T)$$

dove la tensione *V*, la frequenza *f* e la fase φ sono costanti e pari ai valori citati sopra.

Se però si va a vedere qual è l'andamento delle tensioni di un sistema trifase in una situazione reale, si ottiene una situazione diversa e che può cambiare da caso a caso, come mostrato in Figura 2.2 e Figura 2.3 (dati reali forniti dal laboratorio di Misure Elettriche ed Elettroniche dell'Università degli Studi Roma Tre). Ecco, quindi, che quello mostrato in Figura 2.1, con le caratteristiche citate sopra, è solo una situazione ideale e che può essere anche ben lontana dalla realtà.

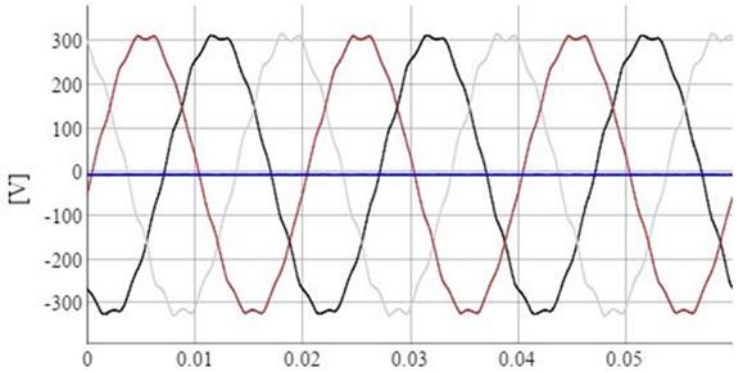


Figura 2.2 - Andamento delle tensioni in un sistema trifase reale lievemente disturbato.

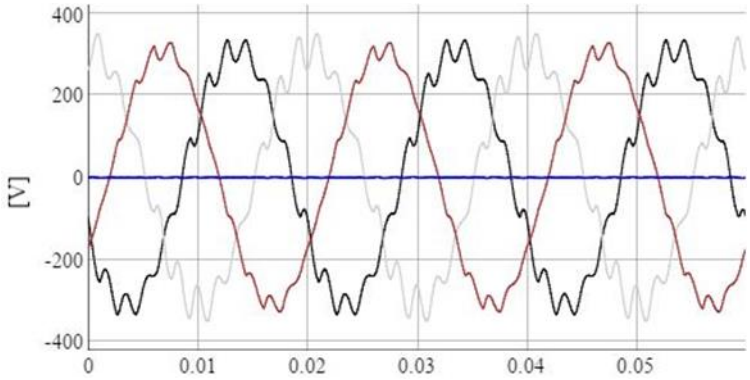


Figura 2.3 - Andamento delle tensioni in un sistema trifase reale fortemente disturbato.

Secondo la normativa UNI EN ISO 9000-2005 la qualità è il "*Grado in cui un insieme di caratteristiche intrinseche soddisfa i requisiti*", quindi si può dire che la "Qualità dell'energia elettrica" di un sistema, o meglio la "Power Quality", esprime di quanto la sua alimentazione reale si avvicina a quella ideale.

- Se la qualità dell'energia della rete è buona, allora ogni carico ad essa connesso funzionerà in modo efficiente e soddisfacente, inoltre, i costi di esercizio, i consumi e quindi le emissioni di anidride carbonica saranno minimi [Schipman].

- Se la qualità dell'energia della rete non è buona, allora i carichi ad essa connessi potrebbero non funzionare correttamente, subire guasti, avere dei tempi di vita più brevi e la loro efficienza energetica potrebbe essere notevolmente ridotta. Inoltre, costi d'esercizio, consumi e di conseguenza l'impatto ambientale saranno elevati e potrebbe essere comunque non possibile eseguire alcune operazioni a causa dei malfunzionamenti [Schipman].

La definizione di "Power Quality" data dalla International Electrotechnical Commission (IEC), nella normativa IEC 61000-4-30, è: "Caratteristiche dell'elettricità in un dato punto del sistema elettrico, stimate rispetto ad un insieme di parametrici tecnici di riferimento". Infatti, per caratterizzare la Power Quality (PQ), differenti indici sono stati definiti, come verrà mostrato più avanti nella sezione riguardante la normativa per la PQ.

2.1 Effetti di una cattiva alimentazione

La presenza crescente di apparati elettronici in tutti i processi energetici e produttivi ha reso i vari utenti della rete elettrica sempre più sensibili a questo discostamento dall'idealità dell'energia. Questo perché una bassa qualità e comportamenti anomali dell'alimentazione possono compromettere il corretto funzionamento della strumentazione elettronica utilizzata e quindi possono condizionare fortemente il normale processo produttivo, soprattutto dove questo è interamente automatizzato.

L'industria avanza impiegando macchinari sempre più veloci, efficienti e produttivi, i quali, però, sono realizzati da apparati che in larga parte risentono enormemente dei disturbi dell'alimentazione, ecco quindi che per cercare di arginare costosi e intollerabili disservizi spesso è necessaria l'adozione di appositi strumenti per la riduzione e compensazione dei disturbi, come gruppi di continuità (UPS), generatori ausiliari, filtri, scaricatori ed altri sistemi che tuttavia non sempre riescono nel loro intento.

Tra le maggiori cause di una cattiva alimentazione in bassa tensione si hanno [Schipman]:

- Potenza reattiva, che va ad aumentare il carico del sistema di alimentazione inutilmente;
- Inquinamento armonico, che provoca ulteriore stress sulla rete elettrica e fa si che gli apparati funzionino meno efficientemente;
- Sbilanciamento del carico. Avere dei carichi molto sbilanciati può provocare un eccessivo sbilanciamento delle tensioni producendo stress agli altri carichi connessi sulla stessa rete, inoltre, porta ad un incremento della corrente di neutro con conseguente aumento della tensione neutro-terra;
- Variazioni veloci di tensione che portano a flicker.

Tra le maggiori conseguenze di una cattiva alimentazione, invece, si hanno [Schipman]:

- Inaspettate mancanze di alimentazione (interruttori di sgancio che si attivano, fusibili che saltano);
- Guasti o malfunzionamenti di dispositivi;
- Surriscaldamento di apparati come trasformatori, motori, ecc… che porta ad un'abbreviazione del loro ciclo di vita;
- Danneggiamento di apparecchiature sensibili come PC, sistemi di controllo delle linee di produzione, ecc…;
- Interferenze su comunicazioni elettroniche;
- Aumento delle perdite di sistema;
- Necessità di sovradimensionare gli impianti per fronteggiare il maggiore stress elettrico, con conseguente aumento dei costi delle strutture e dei costi di esercizio;
- Impossibilità di connettere nuovi apparati alla rete in quanto troppo inquinata.

I disturbi dell'energia elettrica quindi possono avere conseguenze finanziarie significative per diversi utenti e operatori della rete. Data la grande diversità delle cause e per le differenti sensibilità degli apparati è molto difficile stimare correttamente le perdite economiche dovute ad una cattiva alimentazione, tuttavia, nel corso del tempo sono state svolte indagini sul campo, interviste e studi per avere un'indicazione sull'ammontare di queste perdite.

Secondo Baggini [Baggini], i tre settori che risultano particolarmente sensibili ai problemi di Power Quality sono:

- l'economia digitale, ovvero tutte quelle imprese che basano la loro economia sull'elaborazione e sulla memorizzazione dei dati, come ad esempio le industrie di telecomunicazioni e di produzione elettronica;
- i processi di produzione continua, ovvero impianti di produzione che hanno la necessità di lavorare in maniera continua le materie prime, come l'industria della carta, del petrolio e dei metalli;
- i servizi essenziali ed universali, come i servizi idrici, i gasdotti, i servizi di logistica e trasporto merci (come le ferrovie) e tutte le altre industrie manifatturiere.

Considerando la larga diffusione, a tutti i livelli, dell'elettronica di consumo nella società civile, nel primo punto possono essere incluse anche le utenze cosiddette "domestiche".

Secondo uno studio condotto nel biennio 2003-2004 su 62 industrie europee operanti in diversi settori, si è stimato che ogni anno, a causa dei problemi di Power Quality, vengono persi più di 150 miliardi di euro in ambito industriale [Targosz].

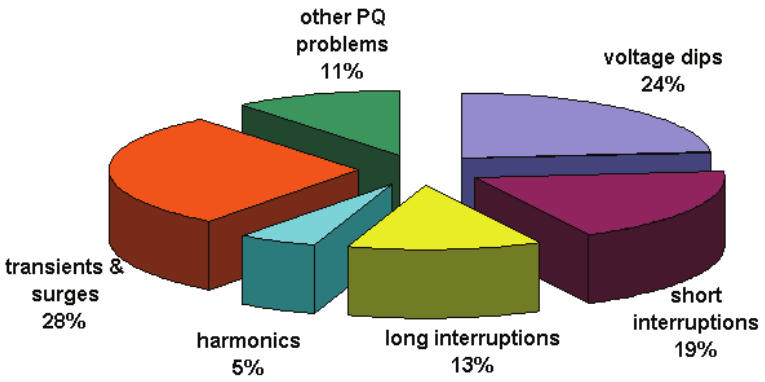


Figura 2.4 – Ripartizione percentuale dei costi dovuti ai problemi di Power Quality rispetto ai vari disturbi (fonte: [Bhattacharyya]).

In Figura 2.4 viene mostrata la ripartizione percentuale delle perdite economiche causate dai disturbi di alimentazione. Si può vedere come la principale fonte di perdita (56% del totale) sia dovuta a buchi di tensione e interruzioni (lunghe e brevi), inoltre, si può notare che anche transitori e sovratensioni hanno una certa rilevanza.

Nello stesso studio viene stimata anche una ripartizione del danno economico su diversi aspetti all'interno di un'industria di produzione, come segue:

- Costo del personale, diventato improduttivo per il flusso di lavoro interrotto;
- "Work in Progress" (WIP), che include il costo delle materie prime coinvolte nella produzione che vengono inevitabilmente perse ed il costo del lavoro aggiuntivo necessario per compensare la produzione persa e ripristinare i danni;
- Malfunzionamento attrezzature, se un dispositivo è interessato, le conseguenze possono essere il rallentamento del processo produttivo ed ulteriori tempi di inattività;
- Danneggiamento apparecchiature, se un apparecchio è colpito, le conseguenze possono essere inservibilità del dispositivo, riduzione del suo ciclo di vita, manutenzione straordinaria, necessità di attrezzatura di rimpiazzo, ecc…;
- Altri costi, dovuti per le sanzioni a causa della mancata consegna o del ritardo nella consegna, per multe ambientali, per lesioni del personale (se presenti), ecc…;
- Costi specifici, che comprendono il costo aggiuntivo dell'energia a causa dell'inquinamento armonico prodotto da dispositivi non lineari e le sanzioni per l'immissione di armoniche nella rete (dove applicabile).

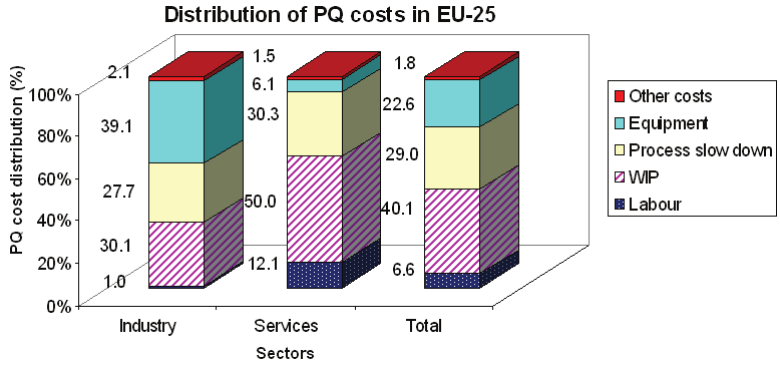


Figura 2.5 – Distribuzione dei costi dovuti alla cattiva PQ per settore (fonte: [Bhattacharyya]).

In Figura 2.5, estratta dall'indagine LPQI [Manson], viene mostrata la ripartizione percentuale dei danni economici subiti nell'industria e nel settore terziario. In questo grafico viene fatta una suddivisione tra costi relativi alle apparecchiature, al "work in progress", al rallentamento dei processi produttivi, al lavoro aggiuntivo necessario e ad ulteriori costi. Come si può vedere gran parte delle perdite sono dovute alle prime tre aree.

In Figura 2.6, invece, sono mostrate le perdite finanziarie stimate per differenti settori industriali dovute solamente ai buchi di tensione [Andersson]. Si può vedere che tutte

le industrie sono sensibili a questo tipo di disturbo e ne risentono maggiormente quelle ad alto contenuto tecnologico come l'industria manifatturiera di semiconduttori.

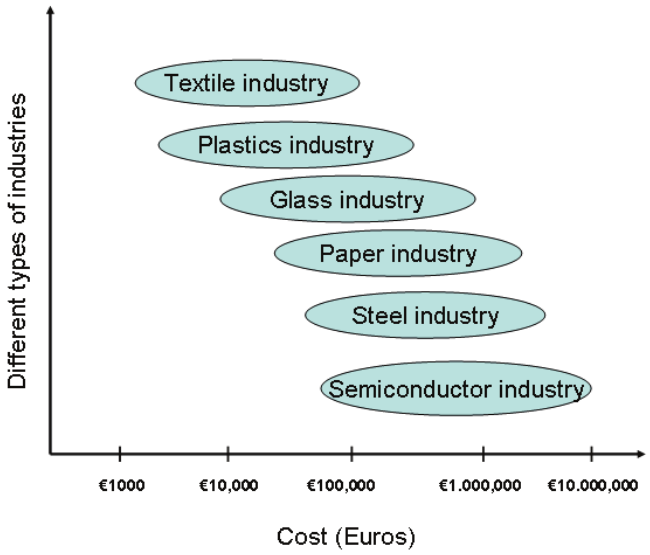


Figura 2.6 – Perdite finanziarie dovute ai soli buchi di tensione in differenti settori produttivi (fonte: [Bhattacharyya]).

In questo contesto il ruolo del monitoraggio della qualità dell'energia diventa fondamentale per acquisire informazioni utili sullo stato di una sottorete elettrica, non solo per individuare e quantificare i disturbi, ma anche per cercare, in qualche modo, di determinare delle responsabilità, soprattutto alla luce dei danni di natura economica, mostrati in questo paragrafo, che possono subire i vari utenti della rete. Inoltre, in alcune situazioni, potrebbe essere possibile prevedere il comportamento della rete ed agire in tempo per prevenire delle anomalie o il peggioramento della situazione.

2.2 Classificazione dei disturbi

Nella Power Quality i disturbi possono essere suddivisi, in base all'entità dello scostamento dalle forme d'onda ideali, nelle due macro-categorie [Bollen] di:

- *Variazioni*, quando si ha a che fare con piccoli scostamenti delle grandezze oggetto d'indagine dall'idealità;
- *Eventi*, quando gli scostamenti dall'idealità sono più ingenti.

Gli standard di compatibilità elettromagnetica (EMC) vengono definiti partendo dal concetto di "livello di compatibilità": si misurano in modo continuativo le variazioni di ogni parametro per ottenerne una distribuzione di probabilità. Il livello di compatibilità è definito come l'intervallo per le variazioni di quel parametro in cui si ha una probabilità del 95% che non esca da tale intervallo. A causa del fatto che si presentano saltuariamente ed in alcuni casi raramente, per gli eventi, invece, occorre rivalutare il concetto di livello di compatibilità, definendolo su basi statistiche, come ad esempio "numero di occorrenze in un anno". Quindi, piccole deviazioni dalla tensione efficace nominale vengono chiamate variazioni di tensione o fluttuazioni di tensione, mentre ampie deviazioni sono chiamate cadute di tensione, sovratensioni o interruzioni.

Si possono considerare *variazioni* i seguenti disturbi:

- Variazioni dell'ampiezza della tensione, causate ad esempio da variazioni di potenza attiva e reattiva assorbita dai carichi;
- Variazioni della frequenza della tensione, rispetto alla frequenza fondamentale relativa alla tensione nominale;
- Variazioni della fase della tensione, cambiamento nel tempo della fase rispetto a quella di riferimento;
- Variazioni dell'ampiezza della corrente, dovuta all'assorbimento non costante dei carichi e che produce una conseguente variazione della tensione di alimentazione;
- Dissimmetria delle tensioni di fase, in genere dovuta alla presenza di carichi monofase che assorbono potenza attiva e reattiva in modo non equilibrato sulle tre fasi. Può portare ad una differenza di ampiezza tra le fasi o ad uno sfasamento diverso da 120° delle stesse;
- Squilibrio delle correnti di fase, fenomeno analogo al precedente ma riguardante le correnti;
- Distorsione della tensione, solitamente dovuta alla presenza di carichi non lineari e trasformatori in saturazione che generano sulla rete componenti a frequenza diversa dalla fondamentale (50 Hz in Italia);
- Distorsione della corrente, fenomeno analogo al precedente ma riguardante le correnti.

Volendo effettuare una classificazione più accurata di questi ultimi due punti, per le distorsioni si può avere:

- Offset DC, o presenza di una componente continua, potrebbe essere causata da un raddrizzatore a semionda malfunzionante e può provocare, in alcuni casi, erosione elettrolitica degli elettrodi di messa a terra;
- Armoniche, sono componenti sinusoidali di tensione (o corrente) che si sovrappongono alla forma d'onda a frequenza fondamentale. Queste componenti hanno una frequenza multipla intera della fondamentale e per ordine h di un'armonica è inteso, appunto, il rapporto tra la sua frequenza di oscillazione f_h e quella della fondamentale f_1, ovvero $h = f_h / f_1$. In genere sono prodotte da carichi e dispositivi non lineari come ad esempio alimentatori switching. Tramite il parametro *Total Harmonic Distortion* (THD), definito come il rapporto (in dB o in percentuale) del valore efficace delle armoniche e quello della fondamentale, è possibile quantificare il tasso di questa distorsione;
- Interarmoniche, sono componenti sovrapposte al segnale che però, rispetto alle armoniche, non hanno una frequenza multipla intera della fondamentale. Queste possono innescare fenomeni di risonanza sulla rete elettrica e producono effetti visibili come lo sfarfallio (o flickering) delle lampade ad incandescenza;
- Notching, sono disturbi periodici della tensione dovuti al normale funzionamento dei rettificatori e convertitori statici tri-fase durante la commutazione della corrente da una fase all'altra. In questo modo vengono introdotte componenti armoniche e non-armoniche a frequenza molto elevata (nel range delle onde radio) che producono interferenze e sono difficili da rilevare con strumentazione ordinaria;
- Fluttuazioni di tensione, variazioni sistematiche o serie di cambiamenti casuali della tensione in cui l'ampiezza non supera i limiti stabiliti dalla normativa ANSI C84.1, ovvero tra il 90% ed il 110% del valore nominale. La normativa IEC 61000-2-1 definisce più tipi di fluttuazioni mentre la IEC 61000-4-15 definisce la metodologia e la strumentazione necessaria per la misurazione dei flicker (sfarfallio delle lampade d'illuminazione provocato dalla fluttuazione della tensione).

Per quanto riguarda gli *eventi*, invece, si possono considerare tali i seguenti disturbi:

- Interruzioni di tensione, quando la tensione ai capi dei terminali di alimentazione è prossima a zero. In particolare, secondo le normative IEEE si può considerare un'interruzione quando il valore della tensione scende al di sotto dell'1% del valore della tensione nominale, mentre per la normativa IEC tale limite viene alzato al 10%. In base alla durata, si hanno interruzioni *brevi* se questa è inferiore al minuto, altrimenti si hanno interruzioni *lunghe*;
- Sottotensioni, si ha una riduzione temporanea dell'ampiezza della tensione seguita da un suo ritorno alla normalità, se il fenomeno ha durata breve si usa il termine inglese "*voltage sag*" o *buco di tensione*;
- Sovratensioni, si ha un aumento temporaneo dell'ampiezza della tensione seguito da un successivo ritorno a regime, anche in questo caso, se il fenomeno ha durata breve (tra 10 ms e 60 s) e l'ampiezza supera il 110% del valore nominale si usa un termine inglese, "*voltage swell*";
- Eventi veloci, o transienti, in questo caso si fa riferimento a generici disturbi di breve durata, solitamente dell'ordine di un periodo del segnale di rete o frazione di questo. Sono definiti anche *eventi transitori*.

Infine, tutti quei fenomeni di disturbo con contenuto spettrale di ampia banda (fino a 200 kHz) che si sovrappongono alla componente a frequenza fondamentale e che non sono classificabili come disturbi armonici o interarmonici, possono essere identificati con il termine inglese "*noise*" (rumore). C'è da dire, inoltre, che la percezione di questo "*rumore*" viene spesso amplificata da connessioni a terra non efficaci.

Nella Tabella 2.1 viene mostrata la classificazione in gruppi dei fenomeni disturbanti secondo la IEC.

Fenomeni condotti	Armoniche, Interarmoniche
	Segnali di controllo
	Fluttuazione della tensione
	Buchi di tensione ed interruzioni
	Dissimmetria della tensione
	Variazioni della frequenza
	Tensioni indotte a bassa frequenza
	Componenti continue nei sistemi in alternata
Fenomeni radiati a bassa frequenza	Campi magnetici
	Campi elettrici
Fenomeni condotti ad alta frequenza	Tensioni o correnti indotte
	Transitori unidirezionali
	Transitori di tipo oscillatorio
Fenomeni radiati ad alta frequenza	Campi magnetici
	Campi elettrici
	Campi elettromagnetici
	Onde continue
	Transitori
Fenomeni di scarica elettrostatica	-----
Impulsi elettromagnetici nucleari	-----

Tabella 2.1 - Classificazione IEC dei fenomeni disturbanti.

Da questa classificazione in gruppi della IEC, prendendo in considerazione la classe dei fenomeni condotti, l'IEEE attraverso lo standard Std.1159.1995 ha proposto una suddivisione più dettagliata dei fenomeni di disturbo riportata in Tabella 2.2. Questa classificazione è oggi quella di riferimento nell'ambito della Power Quality.

Categoria		Contenuto spettrale tipico	Durata tipica	Ampiezza di tensione tipica
Transitori impulsivi	ns	5 ns (tempo di salita)	< 50 ns	
	μs	1 μs (tempo di salita)	50 ns - 1 ms	
	ms	0.1 (tempo di salita)	> 1 ms	
Transitori oscillatori	bassa freq.	< 5kHz	0.3-50 ms	0 - 4 pu (per unità)
	media freq.	5-500 kHz	20 μs	0 - 8 pu
	alta freq.	0.5-5 MHz	5 μs	0 - 4 pu
Variazioni istantanee	sag		0.5-30 periodi	0.1 - 0.9 pu
	swell		0.5-30 periodi	1.1-1.8 pu
Variazioni momentanee	interruzioni		0.5-150 periodi	< 0.1 pu
	sag		30-150 periodi	0.1 - 0.9 pu
	swell		30-150 periodi	1.1 - 1.4 pu
Variazioni temporanee	interruzioni		3 s – 1 min	< 0.1 pu
	sag		3 s – 1 min	0.1 - 0.9 pu
	swell		3 s – 1 min	1.1 - 1.2 pu
Variazioni di lunga durata	interruzioni		> 1 min	0.0 pu
	sottotensioni		> 1 min	0.8 – 0.9 pu
	sovratensioni		> 1 min	1.1 – 1.2 pu
	dissimmetrie		permanente	0.5 – 2%
Distorsioni	DC offset			0 – 0.1%
	armoniche	0 – 100 kHz	permanente	0 – 20%
	interarmoniche	0 – 5 kHz	permanente	0 – 2%
	notching		permanente	
	noise	Banda larga	permanente	0 – 1%
	flutt. tensione	< 25 Hz	intermittente	0.1 – 7%
	var. freq.		< 10 s	

Tabella 2.2 - Suddivisione IEEE dei vari disturbi di PQ.

In Figura 2.7 sono mostrati alcuni esempi di questi disturbi.

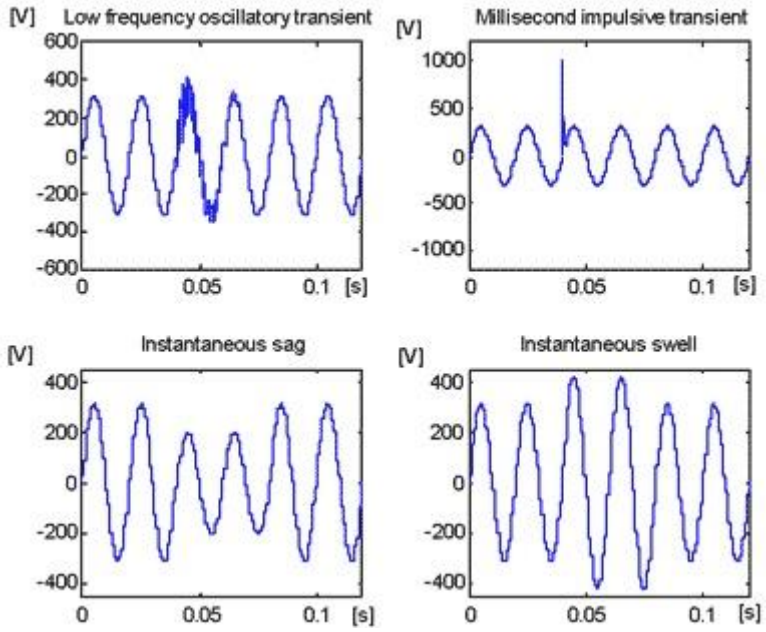


Figura 2.7 - Esempi di alcuni disturbi transitori di tensione (fonte: planetanalog.com).

2.3 Gli Standard per la Power Quality

La documentazione riguardante la parte di compatibilità elettromagnetica della IEC contiene la principale serie di standard internazionali relativi alla Power Quality. Questi standard sono suddivisi in 6 parti principali, ognuna costituita da più sezioni [Bollen]:

- Parte 1: Generale – definizioni di base;
- Parte 2: Ambiente – quantifica i vari livelli di disturbo e descrive l'ambiente e le metodologie atte alla sua quantificazione;
- Parte 3: Limiti – espone i limiti di emissione e immunità per le apparecchiature, in particolare gli standard IEC 61000-3-2 e IEC 61000-3-4 definiscono i limiti per le correnti armoniche, mentre le IEC 61000-3-3 e IEC 61000-3-5 stabiliscono i limiti per le fluttuazioni di tensione;
- Parte 4: Testing e Tecniche di Misurazione – definisce le metodologie per le misurazioni delle emissioni ed i test per l'immunità delle apparecchiature;

- Parte 5: Linee guida per l'installazione e l'attenuazione – indica come prevenire e mitigare i fenomeni d'interferenza elettromagnetica dal progetto alla realizzazione;
- Parte 6: Standard Generici – per tutti quei dispositivi che non dispongono di normative specifiche.

Inoltre esistono anche altri standard per la Power Quality non emessi dalla IEC, come ad esempio l'EN 50160 del CENELEC (Comité Européen de Normalisation Électrotechnique).

La norma italiana CEI EN 50160 "*definisce e descrive le caratteristiche della tensione di alimentazione riguardanti: frequenza, ampiezza, forma d'onda, simmetria delle tensioni trifase. Queste caratteristiche sono soggette a variazioni durante il normale esercizio di un sistema di alimentazione a seguito di variazioni di carico, disturbi generati da determinate apparecchiature e del verificarsi di guasti che sono principalmente dovuti ad eventi esterni*" [CEI].

Queste caratteristiche però non sono destinate in modo esplicito all'individuazione dei livelli di compatibilità elettromagnetica o come limiti di emissione da parte dell'utente, anche se, come dice la stessa normativa, "*dovrebbero essere prese in considerazione*" a tal fine.

Vengono elencate, qui di seguito, alcune delle definizioni contenute nella EN 50160 [CEI], per completezza e raffronto con le normative precedentemente presentate:

a) Frequenza della tensione di alimentazione: *numero di ripetizioni al secondo della componente fondamentale della tensione di alimentazione*;
b) Variazione della tensione: *aumento o diminuzione della tensione normalmente provocato dalla variazione del carico totale del sistema di distribuzione o di una parte di esso*.
c) Variazione rapida della tensione: *variazione rapida singola del valore efficace della tensione tra due livelli consecutivi mantenuti per durate definite ma non specificate*.
d) Fluttuazione della tensione: *serie di variazioni di tensione o una variazione ciclica dell'inviluppo della tensione*.

e) Flicker: *impressione di instabilità della percezione visiva indotta da uno stimolo luminoso la cui luminanza o la cui distribuzione spettrale fluttua nel tempo. Le fluttuazioni di tensione causano variazioni di luminanza delle lampade che, a loro volta, possono determinare il fenomeno visivo chiamato flicker* […].

f) Buco di tensione: *diminuzione improvvisa della tensione di alimentazione compreso tra il 90% e l'1% della tensione dichiarata, seguita da un ripristino dopo un breve periodo di tempo. Convenzionalmente la durata di un buco di tensione è compresa tra 10 ms e 1 minuto* […].

g) Interruzione dell'alimentazione: *condizione nella quale la fornitura è inferiore all'1% della tensione dichiarata* […].

h) Sovratensione temporanea a frequenza di rete: *sovratensione, in una località data, di durata relativamente lunga*.

i) Sovratensione transitoria: *sovratensione oscillatoria o non oscillatoria di breve durata di solito smorzata e con durata di pochi millisecondi o inferiore* […].

j) Tensione armonica: *tensione sinusoidale la cui frequenza è un multiplo intero della frequenza fondamentale della tensione di alimentazione. Le tensioni armoniche possono essere valutate* [...] *globalmente col fattore di distorsione armonica totale (THD) calcolato utilizzando la formula seguente:*

$$THD = \sqrt{\sum_{h=2}^{40} (u_h)^2}$$

con u_h ampiezza relativa della h-esima componente armonica.

k) Tensione interarmonica: *tensione sinusoidale con una frequenza non armonica, cioè la frequenza non è un multiplo intero della fondamentale* [...].

Tale normativa stabilisce anche le caratteristiche dell'alimentazione a bassa tensione, che sono quindi state prese in considerazione per determinare le specifiche riguardanti la scheda di acquisizione che è parte di questo progetto per il monitoraggio della PQ. Principalmente ci si è focalizzati sui seguenti punti:

- Frequenza nominale, "*deve essere di 50 Hz. In condizioni normali di esercizio il valore medio della frequenza fondamentale misurato in un intervallo di 10 secondi deve essere compreso nell'intervallo*":

- 50 Hz ±1% durante il 95% di un anno, 50 Hz +4%/-6% durante il 100% del tempo per sistemi con collegamento sincrono;
- 50 Hz ±2% durante il 95% di una settimana, 50 Hz ±15% durante il 100% del tempo, per i sistemi senza collegamento sincrono;

- Tensione nominale normale U_n, per i sistemi pubblici BT è:
  - U_n = 230 V tra fase e neutro nei sistemi trifase a quattro conduttori;
  - U_n = 230 V tra fasi nei sistemi trifase a tre conduttori;
- Variazioni di tensione, la norma stabilisce che "*in condizioni normali d'esercizio, escludendo le situazioni risultanti da guasti o interruzioni di tensione, durante qualsiasi periodo di una settimana il 95% dei valori efficaci della tensione d'alimentazione, mediato nei 10 minuti, deve essere compreso nell'intervallo* $U_n \pm 10\%$", con U_n intesa come tensione nominale normale.

Questa norma implica dei limiti larghi e poco significativi in quanto, il dover mediare le misure in periodi lunghi, nasconde una serie di variazioni di breve durata. Per questo motivo il sistema di monitoraggio realizzato dal Laboratorio di Misure Elettriche ed Elettroniche del dipartimento di Scienze dell'Università degli Studi Roma Tre, va verso il superamento di tali standard, al fine di produrre dispositivi che possano indagare il fenomeno della PQ anche e soprattutto per scopi scientifici.

Per quanto riguarda invece le modalità di osservazione dei fenomeni relativi alla PQ si fa riferimento ad altri standard, come l'IEEE 1159 assieme alle norme IEC 61000-4-7 e IEC 61000-4-30, specifici per questo scopo e che quindi rappresentano il punto di partenza per la scelta dei requisiti che un sistema di monitoraggio deve soddisfare.

2.4 Il monitoraggio della Power Quality

I disturbi di Power Quality sono fenomeni fisici che, in molti casi, appaiono e scompaiono arbitrariamente, perciò per poterli rilevare non è sufficiente una semplice misurazione di un parametro elettrico, ma è necessario monitorare, analizzare e

registrare vari parametri per un certo intervallo di tempo, a volte anche molto esteso. Per ridurre l'enorme quantità di dati prodotta da monitoraggi di lunga durata vengono imposti dei limiti sulle registrazioni. Se questi limiti vengono superati, lo strumento di monitoraggio registra solo i dati essenziali dell'evento rilevato. Usando algoritmi di aggregazione si riesce quindi a ridurre la quantità di memoria occupata senza perdere informazioni sull'andamento del fenomeno [Sumper].

Per poter rilevare un disturbo sulla rete elettrica è necessario stabilire dei limiti per le grandezze misurabili nella Power Quality. Così, una volta che questi limiti sono definiti, le misurazioni possono essere comparate ed è possibile determinare il livello della qualità dell'energia elettrica sulla rete. Questi limiti, come si è visto nei paragrafi precedenti, possono essere trovati nelle normative, come la EN 50160 che descrive i parametri più importanti per la qualità della tensione sulla rete di distribuzione [Sumper].

C'è da dire che il monitoraggio, da solo, non è la soluzione ai problemi di Power Quality, infatti, per risolvere tali problemi è necessario qualcosa di più che la semplice installazione di monitor per la PQ in un sito, ma sicuramente il monitoraggio gioca un ruolo decisivo in tutto questo. La risoluzione di questi disturbi richiede una prima fase di misurazione e analisi di vari parametri dell'energia e se questa non viene fatta in maniera adeguata potrebbe non essere possibile identificare i problemi e quindi trovare delle soluzioni. Inoltre, una gestione corretta del monitoraggio, può aiutare a minimizzare il costo della risoluzione dei problemi [McGranaghan].

Come si è mostrato dagli studi riportati nei paragrafi precedenti, una cattiva alimentazione può portare a perdite economiche davvero notevoli per gli utenti della rete, ecco quindi, che l'investimento nel monitoraggio può essere ben giustificato poiché fornisce anche maggiori informazioni su [Sumper]:

- manutenzione preventiva e predittiva;
- necessità di attrezzature per la riduzione dei disturbi;
- prestazioni degli apparati;
- sensibilità ai disturbi delle apparecchiature di processo.

2.4.1 Normative per la strumentazione

Nella Comunità Europea, lo standard di riferimento che impone i requisiti che deve soddisfare la fornitura di energia elettrica è l'EN 50160, i cui punti principali sono stati sintetizzati nella Tabella 2.3.

Fenomeno di tensione	Limite accettabile	Intervallo di misurazione	Periodo di osservazione	Tollerabilità percentuale
Frequenza di rete	49.5 Hz to 50.5 Hz	10 s	1 settimana	95%
Variazioni lente	230 V ±10%	10 min	1 settimana	95%
Sags	10 a 1000 volte per anno	10 ms	1 anno	100%
Interruzioni <3 min	10 a 100 volte per anno	10 ms	1 anno	100%
Interruzioni >3 min	10 a 50 volte per anno	10 ms	1 anno	100%
Sovratensioni temporanee	< 1.5 kV	10 ms	-	100%
Sovratensioni transitorie	< 6 kV	-	-	100%
Sbilanciamenti	2%	10 min	1 settimana	95%
Armoniche	8% THD	10 min	1 settimana	95%

Tabella 2.3 - Requisiti principali EN 50160 (fonte: [Shtargot]).

La EN 50160 indica in modo esplicito il numero di armoniche da monitorare per determinare la distorsione armonica totale (THD) della tensione di alimentazione, che comunque deve risultare inferiore all'8%. Poiché impone che nel calcolo di questo coefficiente siano incluse tutte le armoniche fino al 40° ordine, è necessario poter osservare fenomeni oscillatori fino a 2 kHz, ciò si traduce in una frequenza di campionamento minima di 4 kHz. In realtà nella normativa non vengono forniti i limiti corrispondenti alle armoniche di ordine superiore al 25°, poiché "*essi sono generalmente piccoli ma imprevedibili a causa di effetti di risonanza*".

Inoltre, nel più recente standard IEC 61850, viene raccomandato l'impiego di almeno 256 campioni per periodo dell'armonica fondamentale per la registrazione di eventi transitori nei sistemi di energia elettrica. Questo dato dice quindi che in Italia, dove la frequenza nominale della rete è di 50 Hz, bisogna prelevare almeno 12800 campioni al

secondo, imponendo perciò una frequenza di campionamento $f_c \geq 12.8kHz$ e quindi ben superiore ai 4 kHz visti precedentemente.

Infine la normativa IEC 61000-4-30 classifica la strumentazione per la PQ in 3 categorie, definendo i metodi di misurazione ed interpretazione dei dati:

- Classe A: strumentazione di precisione per la verifica di conformità agli standard e per la risoluzione di contenziosi legali; per rientrare in questa classe si deve avere:
    - incertezza del real-time clock interno per la temporizzazione (RTC) non superiore a ±20 ms in presenza di procedura di sincronizzazione tramite segnale radio o GPS, qualora non sia disponibile una sincronizzazione, la tolleranza deve essere migliore di ±1 s nell'arco di 24 h;
    - stima della frequenza sul canale di riferimento ogni 10 s contando il numero di periodi nei 10 s stessi, con un'incertezza non superiore a ±10 mHz nell'intervallo 42.5÷57.5 Hz;
    - misurazione del valore efficace della tensione su 10 cicli di fondamentale a 50 Hz, con un'incertezza non superiore al ±0.1% nell'intervallo compreso tra il 10% e il 150% dell'ampiezza nominale;
    - almeno il 50° ordine per le armoniche e le interarmoniche, in assenza di buchi di tensione;
- Classe S: strumentazione utilizzata per indagini statistiche o accertamenti di PQ, possibilmente riferita ad un ristretto insieme di parametri; deve avere:
    - incertezza del suo RTC non superiore a ±5 s per un periodo di 24 h;
    - stima della frequenza come per la Classe A, a differenza dell'incertezza che non deve essere superiore ai ±50 mHz nell'intervallo 42.5÷57.5 Hz;
    - misurazione dell'ampiezza della tensione analogamente alla Classe A, con un'incertezza non superiore a ±0.5% nell'intervallo compreso tra il 20% e il 150% dell'ampiezza nominale;

- è sufficiente il 40° ordine per le armoniche (come nella EN 50160), anche in presenza di buchi di tensione; il calcolo delle interarmoniche è lasciato al costruttore.

- Classe B: ha perso di interesse in quanto la normativa è propensa all'abbandono della stessa; questa classe è stata introdotta per mantenere in vita strumentazione preesistente alla normativa, lasciando la specificazione delle caratteristiche al costruttore.

Come si può vedere, la classificazione appena fatta, non fornisce tanto informazioni a proposito della progettazione dello strumento, ma definisce solo dei requisiti prestazionali che lo strumento deve possedere. In questo progetto l'hardware realizzato cerca di andare oltre tutte queste normative, proponendo un sistema che abbia validità e fruibilità scientifica piuttosto che legale.

2.4.2 I sistemi di monitoraggio

Un sistema di monitoraggio per la Power Quality è un qualcosa capace di ricavare informazioni utili sulla rete elettrica sotto esame, mediante operazioni di raccolta, analisi ed interpretazione di dati di misura grezzi [Dugan].

Originariamente si aveva che, il processo di raccolta avveniva in modo automatizzato con misurazioni continuative di tensione e corrente effettuate su un lungo intervallo temporale, mentre, il processo di analisi ed interpretazione dei dati veniva svolto manualmente dall'essere umano. Oggigiorno, grazie all'enorme ed incessante sviluppo dell'elettronica, tra l'altro sempre più integrata e miniaturizzata, si ha la possibilità di integrare nella strumentazione capacità di calcolo ed elaborazione del segnale e quindi produrre apparecchiature di misura con funzionalità di analisi ed interpretazione automatica dei risultati.

In commercio esistono svariati apparati in grado di effettuare misurazioni di Power Quality, realizzati anche da marchi prestigiosi nell'ambito della strumentazione di misura come Fluke Corporation, HT Instruments, Yokogawa, AEMC Instruments, Dranetz e National Instruments. Ogni casa ha nel suo listino diversi modelli, ognuno con caratteristiche diverse e che quindi sono in grado, chi in classe A, chi in classe S o B, di soddisfare i requisiti per le misurazioni di Power Quality. Questi strumenti, però, solitamente sono pensati per essere utilizzati da un operatore ed anche se possono

effettuare analisi per lunghissimi periodi di tempo, rimangono apparati "portabili" e non sono pensati per essere installati in maniera permanente (o quasi) nei siti di monitoraggio. Inoltre, hanno alcune caratteristiche configurabili ma altre sono fisse e non modificabili, perciò anche se rispettano le normative possono presentare alcune difficoltà nell'andare oltre queste ultime, e mentre questo non è un problema se si vuole operare in ambito legale potrebbe esserlo in ambito scientifico. Oltre a questi "svantaggi" per l'ambito scientifico, hanno anche la caratteristica di essere molto costosi, soprattutto le versioni più configurabili che possono superare anche le decine di migliaia di euro.

Considerando l'obiettivo di avere una rete di monitoraggio fissa estesa geograficamente sul territorio e con più punti di osservazione possibili, si ha che il costo elevato e la limitata configurabilità rende l'uso dei dispositivi in commercio poco adatto al caso.

Il laboratorio di Misure Elettriche ed Elettroniche del dipartimento di Scienze dell'Università Roma Tre è già da diversi anni che si occupa del monitoraggio geografico della Power Quality e nel tempo ha sviluppato una strumentazione proprietaria che è sufficientemente configurabile e comunque adatta per l'ambito scientifico. Questa strumentazione è stata presa come punto di partenza per il lavoro svolto durante il periodo di dottorato, ma presentando molti punti deboli è stata man mano migliorata e si è evoluta fino a diventare un'apparecchiatura completamente diversa da quella di partenza.

2.4.2.1 Il vecchio sistema di monitoraggio del MeaLab

Il laboratorio di Misure Elettriche ed Elettroniche (MeaLab) è dal 2004 che è impegnato attivamente nel monitoraggio della qualità dell'energia elettrica in bassa tensione. Ha avuto vari accordi con enti esterni e tra questi uno dei più importanti (per la Power Quality) è stato quello con la società di telecomunicazioni Telecom Italia S.p.A.. Obiettivo di questo accordo era quello di capire e studiare la qualità dell'energia elettrica che arrivava nelle centrali di Telecom Italia del comune di Roma, per cercare di comprendere se i problemi che si verificavano spesso sulle apparecchiature di telecomunicazioni erano legati o meno ai disturbi sulla rete elettrica. Questo ha permesso al laboratorio di installare i dispositivi all'interno di 4 cabine di trasformazione da media tensione (MT) a bassa tensione (BT) di proprietà di Telecom

Italia, che alimentavano interamente le 4 centrali sovrastanti le cabine. In Figura 2.8 è mostrata la posizione sul territorio di Roma dei 4 siti di monitoraggio.

Figura 2.8 – Dislocazione dei siti di monitoraggio del MeaLab sul territorio di Roma.

L'installazione delle apparecchiature all'interno dei locali messi a disposizione da Telecom Italia, oltre a permettere l'accesso alla rete elettrica trifase direttamente sull'allaccio principale, ha anche reso disponibile, in ogni sito, la presenza di:

- una linea telefonica per il traffico voce;
- una connessione dati a banda larga (ADSL);
- un segnale metrologico di sincronizzazione a 2048kHz, proveniente dall'Istituto Nazionale di Ricerca Metrologica (I.N.Ri.M.) di Torino e che presenta un'elevata accuratezza di una parte su 10^{-14}.

La strumentazione di analisi della Power Quality realizzata ed utilizzata dal MeaLab all'inizio del mio periodo di dottorato era già il risultato di varie integrazioni e

miglioramenti effettuati dal personale del laboratorio durante gli anni precedenti. Verrà mostrata adesso una schematizzazione di questo "vecchio" sistema (Figura 2.9) ed a seguire una descrizione dei vari blocchi.

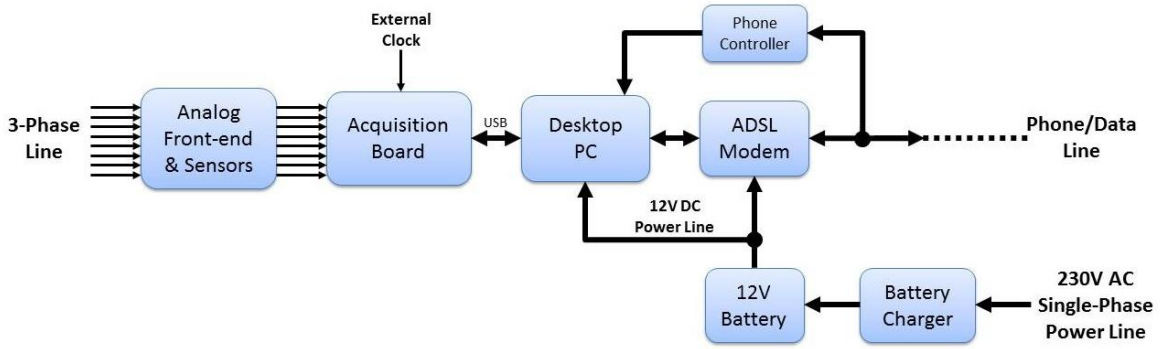


Figura 2.9 – Schema a blocchi del "vecchio" sistema di monitoraggio del MeaLab.

Partendo da sinistra, si ha un primo stadio analogico di condizionamento, necessario per riportare i valori dei segnali da monitorare in una forma ed in un range adatto al successivo stadio di conversione. In Figura 2.10 viene mostrato uno schema di questo primo stadio.

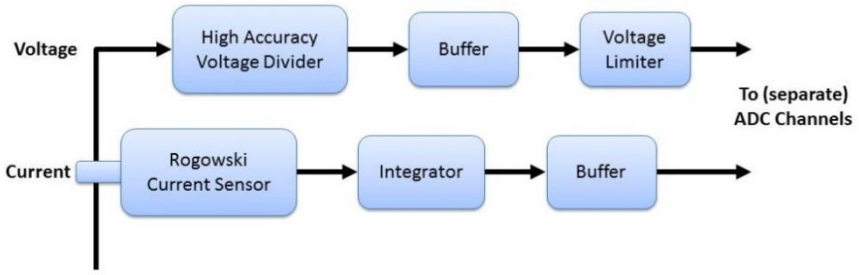


Figura 2.10 - Condizionamento del segnale analogico.

Le tensioni delle tre fasi R-S-T e del neutro N vengono ridotte di 100 volte da un partitore resistivo (uno per fase) ad elevata accuratezza (rapporto 100:1 e incertezza sul valore di ±0.01%) per poter essere adattate al range di funzionamento del convertitore analogico digitale presente nello stadio successivo. Inoltre, si ha anche un limitatore di tensione che blocca eventuali tensioni troppo elevate ed un buffer per l'adattamento d'impedenza.

Per le correnti, viene operata una trasduzione mediante l'impiego di bobine di Rogowski (Figura 2.11). Queste consistono in un cavo conduttore avvolto elicoidalmente su di un supporto flessibile per formare una bobina, un'estremità del solenoide così formato viene fatta passare all'interno del solenoide stesso in modo da essere riportata all'origine della bobina e formare così un dispositivo libero ad una

estremità che risulta facilmente avvolgibile intorno ad un cavo o ad un flusso di corrente che si intende misurare sfruttando la legge di Ampére. Il dispositivo elettrico così creato ha la caratteristica di essere non invasivo sul circuito rispetto ad altri sensori come i trasformatori di corrente ed inoltre non essendo avvolto su un'anima di ferro presenta bassa induttanza che lo rende efficace nelle misurazioni di correnti alternate, impulsive e che variano velocemente nel tempo e presenta un'elevata linearità anche se sottoposto a grandi correnti (kA). Il Rogowski coil fornisce un segnale di tensione che è proporzionale alla derivata della corrente che scorre nel cavo, quindi è necessario uno stadio di integrazione successivo per ottenere un segnale in tensione direttamente proporzionale a quello in corrente.

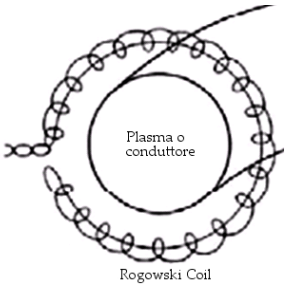


Figura 2.11 - Struttura di un Rogowski coil (fonte: Wikipedia).

Successivamente si ha la scheda di acquisizione che è stata progettata e realizzata dal MeaLab. In Figura 2.12 si ha uno schema di principio del funzionamento di questa scheda, una descrizione più completa si può trovare in [Caciotta].

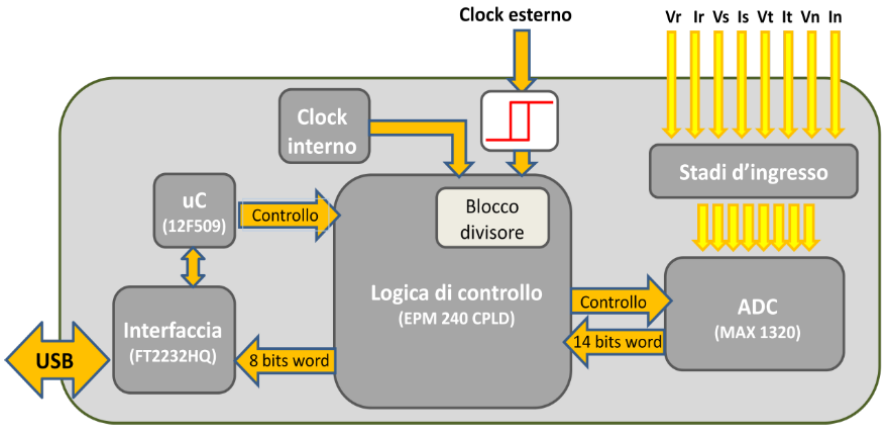


Figura 2.12 - Schema di principio della scheda di acquisizione realizzata dal laboratorio.

In figura vengono mostrati anche gli stadi d'ingresso visti prima.

I componenti principali di questa scheda sono:

- **Convertitore analogico-digitale (ADC)** ad approssimazioni successive (SAR) con 14 bit di risoluzione ed in grado di campionare simultaneamente i segnali sugli 8 canali d'ingresso in quanto dotato di 8 sample&hold che si aprono e chiudono contemporaneamente. Inoltre ha un range permesso per i segnali di ingresso di ±5V e frequenza di campionamento fino a 250kS/s per canale;
- **Logica di controllo sviluppata su CPLD Altera**, si occupa della gestione dei segnali di controllo dell'ADC e dei dati convertiti in uscita da questo. Può gestire le temporizzazioni da clock locale (interno) o da quello esterno collegato al segnale metrologico presente nei siti di monitoraggio con frequenza di 2048kHz.
- **Interfaccia di comunicazione USB con il PC,** tramite un convertitore USB/seriale (FT2232HQ) vengono virtualizzate sul computer due porte seriali per una semplice gestione della comunicazione bidirezionale della scheda. Inoltre l'intera scheda di acquisizione è alimentata tramite questa interfaccia;
- **Microcontrollore,** collegato all'interfaccia USB (tramite il convertitore), ha il compito di avviare le acquisizioni o settare la frequenza di campionamento comandando la CPLD.

Tornando allo schema generale di Figura 2.9, si ha un Personal Computer desktop con sistema operativo Windows 2000 e su cui gira un software di controllo e calcolo dei parametri sviluppato ad hoc dal laboratorio in Visual Basic 6. In Figura 2.13 è mostrata una schermata di questo software che ha le seguenti funzioni:

- Controllo e gestione della comunicazione con la scheda di acquisizione (tramite interfaccia USB) che comprende:
    - settaggio frequenza di campionamento;
    - impostazione buffer di campioni da acquisire;
    - richieste di campionamento ad intervalli regolari settabili;
    - ricezione dei campioni delle tensioni e delle correnti.
- Calcolo dei parametri di Power Quality effettuato sui campioni acquisiti tramite un algoritmo sviluppato dal MeaLab. I parametri stimati sono:
    - Valori efficaci di tensioni e correnti di fase;
    - Frequenza della fondamentale;
    - Ampiezza e fase di ogni singola armonica di tensione e corrente;
    - Potenze assorbite (apparente, attiva, reattiva, armonica);

- Fattore di potenza sulle varie fasi;
- Sequenza diretta, inversa e omopolare del sistema trifase.

- Salvataggio, per ogni acquisizione, di un file di testo sulla memoria locale, contenente i vari parametri calcolati ed il timestamp dell'acquisizione.

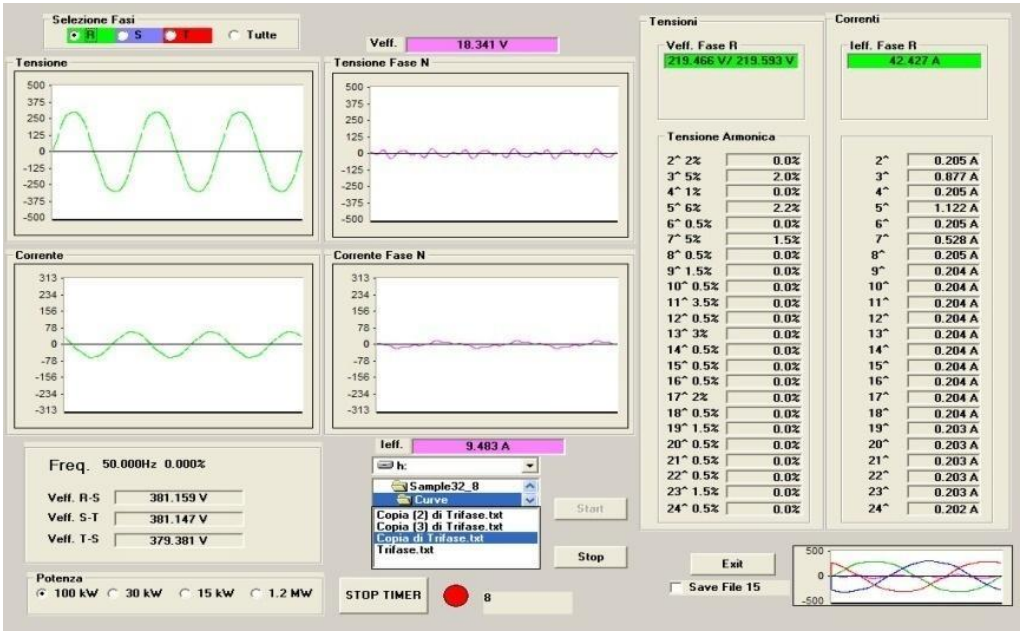


Figura 2.13 - Interfaccia del software di gestione della scheda.

Nonostante le varie possibilità di settaggio per l'acquisizione, solitamente venivano campionati 3 periodi della 50 Hz (ovvero 60 ms) ad una frequenza di circa 25 kHz, producendo ben 1500 campioni per ogni canale. Le acquisizioni venivano lanciate allo scoccare di ogni minuto rispetto all'orologio di sistema del pc.

La presenza del controller telefonico permetteva, tramite una semplice telefonata, di conoscere lo stato del sistema (acceso, spento, batteria carica/scarica) ed eventualmente di intervenire accendendo, spegnendo o riavviando il computer.

Il sistema di alimentazione dell'apparecchiatura era costituito da una batteria 12V 60Ah ed un caricabatteria continuamente connesso alla rete elettrica ed alla batteria stessa. In questo modo, anche durante le interruzioni di alimentazione, lo strumento rimaneva acceso (autonomia fino a 36h) per poter essere in grado di monitorare lo stato della rete al ritorno dell'energia elettrica.

Inoltre, il sistema era collegato ad internet grazie ad un modem ADSL e costantemente connesso su una rete privata virtuale (VPN) costituita utilizzando il software di Hamachi. Questo era necessario per far si che, a cadenze più o meno prefissate, ci si

potesse collegare manualmente da remoto ai pc dei punti di monitoraggio per poter scaricare i dati acquisiti e liberare memoria sui siti locali.

In Figura 2.14 sono mostrate due immagini del sistema di monitoraggio installato in uno dei siti di Roma effettuate durante un intervento di manutenzione (il che giustifica la presenza di mouse, tastiera e monitor sul tavolo, oltre ai componenti descritti precedentemente).

Figura 2.14 - Immagini del sistema di monitoraggio in uno dei siti di Roma durante un'operazione di manutenzione.

L'infrastruttura realizzata

Obiettivo di questo lavoro di dottorato è stato quello di realizzare un'infrastruttura adatta al monitoraggio geografico della Power Quality in bassa tensione su **larga scala**, ovvero un'infrastruttura che possa avere numerosi punti di osservazione (anche centinaia o più) distribuiti ovunque sul territorio e che abbia la capacità di gestirli e controllarli tutti da remoto.

Come si è già detto nel capitolo precedente, attualmente in commercio esistono numerosi dispositivi per misurazioni di PQ ma hanno lo svantaggio di essere molto costosi e spesso poco configurabili, il che non li rende molto adatti ad un monitoraggio su centinaia di siti e per ambito scientifico. Avendo già il laboratorio di Misure Elettriche ed Elettroniche un certo "know-how" sull'argomento del monitoraggio in ambito scientifico ed avendo già, quindi, della strumentazione realizzata ad hoc, seppur con diversi limiti e punti deboli, si è pensato di utilizzare questa come punto di partenza.

Quindi, l'obiettivo di questo lavoro è stato quello di migliorare notevolmente la strumentazione per il monitoraggio della Power Quality a disposizione del laboratorio (descritta nel capitolo precedente), cercando di risolvere tutti i suoi punti critici per l'applicabilità in un progetto di monitoraggio della PQ distribuito geograficamente su larga scala.

Le limitazioni principali del "vecchio" sistema di monitoraggio del MeaLab sono:

a) **Acquisizioni con elevata incertezza sull'istante preciso di inizio campionamento**, sia localmente sia rispetto a file con stesso timestamp provenienti da siti differenti. Il vecchio meccanismo di funzionamento prevede che sia il computer (tramite il software di controllo) a far partire l'acquisizione dei segnali elettrici inviando, ogni volta, un comando alla scheda di acquisizione. Inviando questo comando allo scoccare di ogni minuto, rispetto all'orologio di sistema, si ha una forte incertezza temporale sull'istante esatto di

quando la scheda riceve il comando e quindi inizia effettivamente a campionare la rete. Questa incertezza temporale è dovuta principalmente a:

- Lentezza dell'hardware del computer nell'eseguire i comandi, i computer usati erano stati scelti maggiormente per i loro bassi consumi energetici e non per le loro performance di calcolo quindi erano lenti e poco reattivi;
- Lentezza del Sistema Operativo, il software di controllo, girando su Windows, doveva condividere le risorse hardware (già limitate) con i numerosi processi in esecuzione del S.O., ritardando quindi l'esecuzione dei comandi uscenti dal software;
- Deriva dell'orologio di sistema. Per riconoscere lo scoccare del minuto veniva fatto il polling continuo sull'orologio di sistema, questo però ha una sua deriva che aumenta continuamente nel tempo. Sebbene ci fosse un'operazione pianificata in Windows di sincronizzare ogni giorno l'orologio con un server NTP (Network Time Protocol), spesso questa operazione falliva e l'orologio di sistema continuava la sua deriva senza venire corretto.

Nel complesso questa incertezza sull'istante preciso di inizio campionamento poteva tranquillamente variare da decine di millisecondi a diversi minuti, ma soprattutto era diversa da acquisizione ad acquisizione dato che i processi in esecuzione nel S.O. e la deriva dell'orologio potevano cambiare continuamente. Questo, inoltre, portava anche ad un disallineamento temporale dei vari record acquisiti nello stesso minuto ma su siti differenti.

b) **Costo del singolo punto di monitoraggio.** Sebbene il costo di un apparato sia inferiore a quello di molti strumenti per PQ in commercio, è comunque un costo elevato se si pensa di distribuire il monitoraggio geograficamente su **larga scala** (il costo complessivo di un punto di monitoraggio del vecchio sistema, esclusi i sensori di corrente, si aggirava sui 2000 €);
c) **Bassa affidabilità dei sistemi,** che spesso richiedevano interventi di manutenzione sul sito e che, inoltre, non prevedevano la possibilità di una manutenzione da remoto;
d) **Dimensioni delle apparecchiature,** che risultavano un po' ingombranti nei locali dove dovevano essere installate (tutta la strumentazione poteva essere inserita in un box metallico di circa 80 dm^3);

e) **Consumi,** potendo ridurre i consumi del punto di misurazione si sarebbe potuta ridurre, a parità di autonomia energetica, la capacità della batteria in tampone e di conseguenza peso ed ingombri del sistema;

f) **Autonomia di memorizzazione e sicurezza dei dati,** spesso si verificavano problemi sulla VPN rendendo impossibile il recupero dei dati da remoto che, quindi, venivano accumulati nell'HD del computer che man mano si riempiva. Inoltre, in caso di crash o anomalie sul pc questi dati potevano corrompersi e divenire inutilizzabili;

g) **Assenza di un sistema per gestire l'intera infrastruttura di monitoraggio con tutti i suoi siti.** Come già detto, i dati venivano scaricati manualmente da ogni probe tramite una VPN ed in caso di problemi era necessario recarsi di persona sul sito, in quanto, non era prevista manutenzione da remoto se non per il controller telefonico che però spesso non era sufficiente.

Dopo una lunga serie di step evolutivi, che man mano hanno migliorato e risolto varie problematiche, sia hardware che software, si è arrivati alla versione finale mostrata con uno schema semplificativo in Figura 3.1.

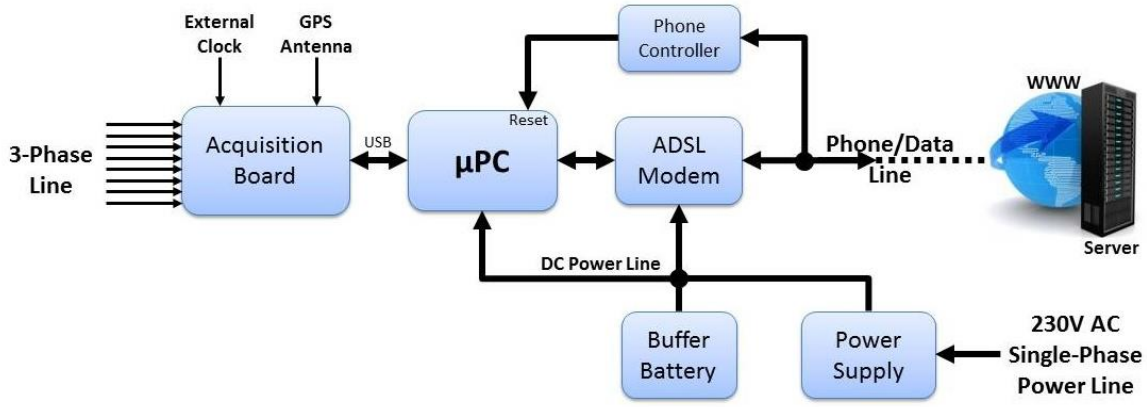


Figura 3.1 - Schema a blocchi dell'Infrastruttura realizzata.

Da questa figura si può già avere un'idea generale del funzionamento del punto di monitoraggio: i segnali elettrici vengono campionati dalla scheda di acquisizione e trasmessi ad un micro-computer, qui vengono elaborati per poi essere trasferiti in un server centrale utilizzando una connessione dati. Poi si ha un sistema di alimentazione continuo, per dare energia a tutto l'apparato ed un controller telefonico da utilizzare in caso di anomalie. Oltre a questo, si ha tutta una parte software molto elaborata ed una parte dedicata alla gestione dell'intera infrastruttura.

Per descrivere con chiarezza l'intera infrastruttura realizzata, verrà esposta nei paragrafi successivi considerando separatamente i suoi 3 aspetti principali:

- la parte *Hardware*, che si occupa dell'acquisizione dei segnali della rete elettrica e della memorizzazione fisica dei dati;
- la parte *Software*, che si occupa dell'elaborazione dei segnali acquisiti, della stima dei vari parametri e della visualizzazione di questi sul web;
- la parte dedicata al *Sistema di Gestione e Controllo*, che permette di gestire da remoto l'intera infrastruttura: sia i vari punti di monitoraggio che il server.

3.1 L'Hardware

In questo paragrafo verrà descritto l'hardware dell'intera infrastruttura realizzata (mostrato in Figura 3.1), analizzandolo un blocco alla volta.

3.1.1 La scheda di acquisizione

Il primo blocco da esaminare è quello della scheda di acquisizione che si occupa di inserirsi sul sistema trifase da monitorare, per poi campionare e digitalizzare i segnali elettrici e passarli al blocco successivo per l'elaborazione. In Figura 3.2 viene mostrato uno schema funzionale di questo primo blocco.

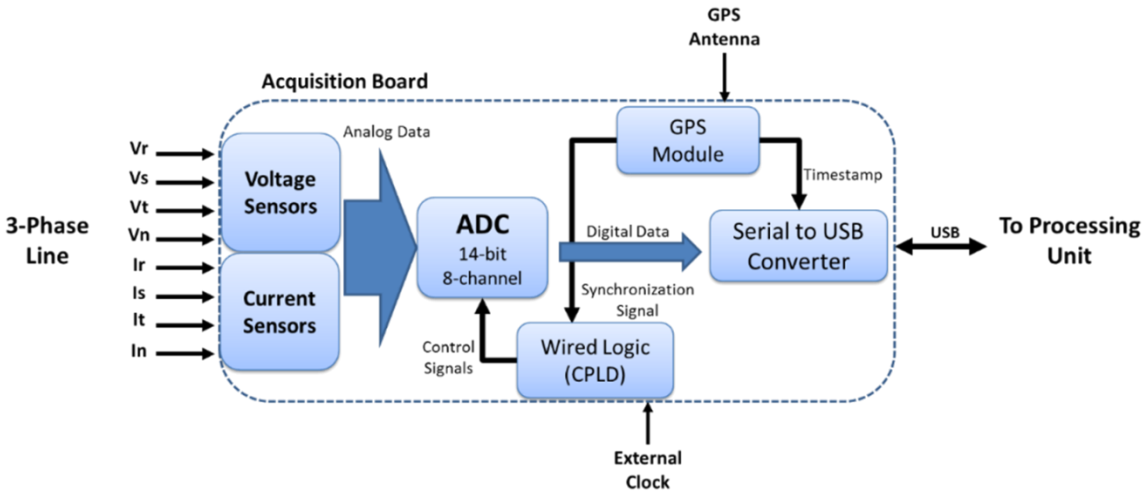


Figura 3.2 - Schema di funzionamento dell'Acquisition Board.

I punti di monitoraggio hanno il compito di acquisire i valori di tensioni e correnti di un sistema trifase a 4 conduttori (fasi R-S-T e neutro), quindi, la scheda ha la capacità di acquisire simultaneamente da 8 canali analogici.

Per quanto riguarda il primo *stadio di condizionamento ed interfacciamento con la rete* si è mantenuto lo stesso principio, già ben studiato e collaudato dal laboratorio, utilizzato nel vecchio sistema, descritto nel paragrafo 2.4.2.1 e di cui viene riportato lo schema in Figura 3.3.

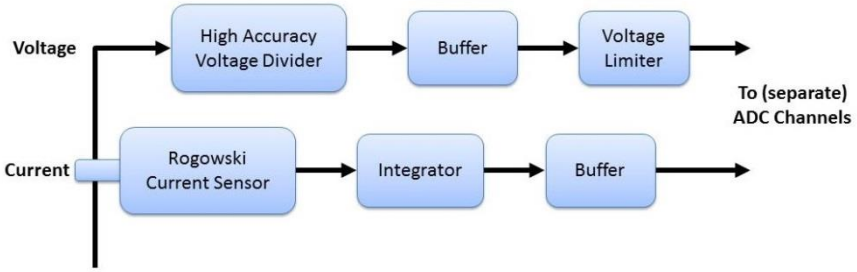


Figura 3.3 – Condizionamento del segnale analogico.

Quindi, il sensing delle tensioni viene fatto con dei partitori resistivi ad elevata accuratezza e con rapporto di partizione 100:1, che permettono un'acquisizione senza distorsioni e garantiscono la compatibilità dei segnali con lo stadio di conversione successivo per tensioni sinusoidali fino ad 1kV picco-picco. Per le correnti, invece, vengono utilizzate delle bobine di Rogowski, seguite da un integratore, che riportano in tensione l'informazione sulle correnti. Essendo questa parte già stata trattata nel testo

ed anche in varie pubblicazioni si rimanda al paragrafo 2.4.2.1 ed a [Caciotta2] per ulteriori chiarimenti ed informazioni.

Una volta che i segnali analogici sono stati condizionati e la loro dinamica resa compatibile con il range permesso dal convertitore analogico digitale vengono, appunto, mandati sugli 8 canali d'ingresso dell'*ADC*. Il convertitore ad approssimazioni successive SAR utilizzato (MAX1320 della società Maxim Integrated [MAX1320]) ha i seguenti punti di forza principali:

- *8 canali analogici indipendenti a* ***campionamento simultaneo***, il dispositivo presenta "on-chip" 8 canali separati, ognuno con un proprio circuito di Track & Hold ed è quindi in grado di campionare tutti i canali nello stesso istante (con un'incertezza massima tra canali di 50 ps);
- *Frequenza di campionamento fino a 250ksps* abilitando tutti i canali;
- *14 bit di risoluzione*;
- 77dB di SNR a 100kHz;
- Range permesso per i segnali in ingresso duale e simmetrico di ±5V.

Aspetto molto importante di questo ADC è proprio quello di essere in grado di campionare i segnali sui vari ingressi esattamente nello stesso istante, in modo da non perdere informazioni di fase sui segnali elettrici.

Il *modulo GPS*, inserito nella scheda e dotato di antenna esterna, è fondamentale per due funzioni principali:

- Fornire il TimeStamp da associare ai dati acquisiti;
- Erogare un segnale in grado di sincronizzare la scheda.

Il modulo utilizzato è il *Resolution-T* della Trimble [Resolution-T] ed ha un'architettura studiata appositamente per l'impiego in circuiti in cui è necessaria un'accurata sincronizzazione temporale. Questo modulo, allo scoccare di ogni secondo, fornisce un segnale impulsivo (PPS - Pulse Per Second) sincronizzato con il GPS con un'incertezza massima di 15ns ed una stringa di testo con tutte le informazioni riguardanti il timestamp, lo stato del modulo ed i satelliti a cui è agganciato.

La stringa di testo in uscita dal Resolution-T viene mandata, tramite una connessione seriale, ad un microcontrollore che per semplificare non è stato inserito in Figura 3.2 perchè può essere considerato come parte del "GPS Module". Questo microcontrollore (un PIC ad 8 bit della società Microchip Technology Inc.) ha il compito di leggere il contenuto della stringa per capire se il modulo GPS, e quindi il suo orologio interno, è sincronizzato con i satelliti. Solo in tal caso, abilita il passaggio del segnale PPS diretto verso la CPLD. Inoltre, il PIC ha la funzione di ri-trasmettere il timestamp contenuto nel testo al convertitore seriale/usb.

Le temporizzazioni e i segnali di controllo dell'ADC sulla scheda sono gestiti da *logica programmabile* per garantire che i ritardi di propagazione sui segnali siano minimi e soprattutto sempre stabili nel tempo, cosa che non avverrebbe nel caso ad esempio si usasse un microcontrollore. Per la temporizzazione delle acquisizioni, scandite dalla CPLD (Complex Programmable Logic Device), si utilizza un clock metrologico ("external clock") che è presente nelle centrali di telecomunicazioni di Telecom Italia, per cui nei siti dove questi dispositivi sono stati installati. Questo clock, proveniente dall'Istituto Nazionale di Ricerca Metrologica (I.N.Ri.M.) di Torino, presenta una frequenza di oscillazione di 2048 kHz, con un'accuratezza elevatissima di una parte su 10^{-14} e ciò assicura un'altissima stabilità della frequenza di campionamento. La scheda, inoltre, è dotata di un oscillatore locale, avente la stessa frequenza, in modo da poter continuare a funzionare anche in assenza del clock metrologico.

Attualmente la CPLD è stata programmata per pilotare l'ADC in modo che possa eseguire 1500 acquisizioni consecutive (contemporaneamente sugli 8 canali) ad una frequenza di circa 25 kHz, ogni minuto. Questa frequenza permette di rispettare gli standard sulla strumentazione visti nel paragrafo 2.4.1. La frequenza di campionamento è ottenuta, a livello hardware all'interno della CPLD, dividendo il clock a 2048 kHz per 82. In Figura 3.4 viene mostrata una parte del circuito interno della CPLD in cui si può vedere come avviene questa divisione in logica programmabile. E' stato utilizzato il "carry out" di un contatore a modulo 82 (quello in alto) per generare la frequenza di 25 kHz a partire dal segnale a 2048 kHz ed il "carry out" di un contatore a modulo 1500 (in basso) per bloccare le acquisizioni una volta raggiunti i 12000 campioni (1500 campioni per 8 canali). Raggiunto questo limite tutto viene azzerato ed arrestato finchè non arriva l'impulso di "Conversion Start" (CS) che fa ripartire il nuovo burst di 1500

acquisizioni. Il segnale CS (in alto a sinistra) viene generato ogni minuto da un altro contatore (non mostrato in figura) oppure quando arriva il PPS proveniente dal modulo GPS (che va a resettare anche il contatore del minuto).

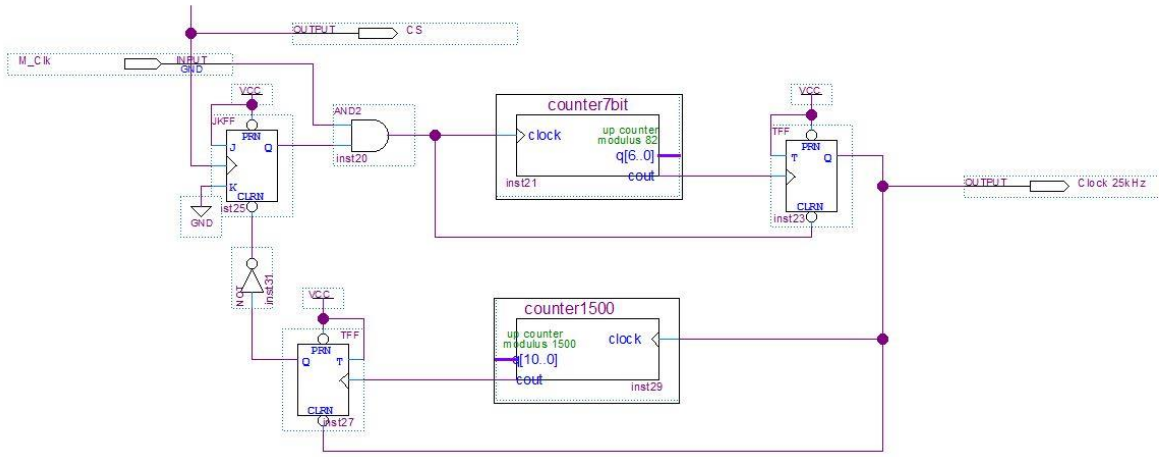


Figura 3.4 - Generazione della frequenza di campionamento all'interno della CPLD.

Un aspetto molto importante che è stato implementato nella CPLD è quello di resettarsi e risincronizzarsi ogni volta che gli arriva il segnale PPS di sincronizzazione fornito dal modulo GPS. In Figura 3.5 è visibile lo schematico della parte che si occupa di questa operazione.

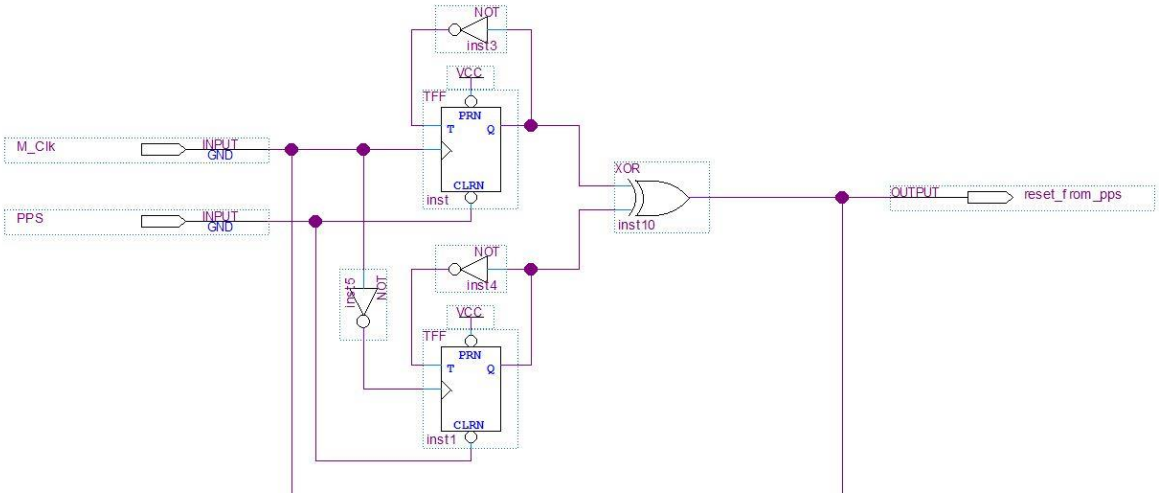


Figura 3.5 - Parte della logica interna alla CPLD che si occupa della sincronizzazione della stessa all'arrivo del PPS.

Grazie all'utilizzo di due flip-flop di tipo T (Toggle), in questa apparentemente banale configurazione, si è riuscito a generare un segnale in grado di resettare il conteggio del minuto e far ripartire tutta la scheda con un ritardo massimo sul PPS pari 244 ns e cioè metà periodo del clock metrologico (M_Clk). Infatti, in questa disposizione, i due flip-flop sono in grado di rilevare entrambi i fronti del clock a 2048 kHz, facendo ripartire

il campionamento al primo fronte rilevato. Per la versione completa dello schematico si rimanda in Appendice dove è stato allegato per intero.

Infine, l'ultimo blocco della Figura 3.2 rappresenta un *convertitore seriale/usb*, più nello specifico il modulo FT2232H della società Future Technology Devices International (FTDI). Il modulo usb in questione implementa due porte com virtuali (VCP) che possono essere facilmente riconosciute e gestite da un sistema operativo. Una porta viene usata per comunicare al blocco successivo il timestamp, mentre l'altra viene utilizzata per trasmettere i dati digitalizzati dall'ADC (questi ultimi vengono ricevuti parallelamente e trasmessi serialmente dal modulo). Inoltre, il modulo ha il compito di alimentare tutta la scheda di acquisizione, prelevando energia dalla connessione usb e fornendo le tensioni necessarie ai vari componenti della scheda.

Quindi, ricapitolando, la scheda è in grado di acquisire simultaneamente i segnali di tensioni e correnti del sistema trifase (R-S-T-N) ad una frequenza di campionamento con elevata stabilità temporale, grazie all'impiego di logica programmabile e di un segnale di clock metrologico ad elevata accuratezza presente nelle centrali di telecomunicazioni di Telecom Italia. I dati così acquisiti, corredati da un timestamp, vengono convertiti serialmente e trasmessi al blocco successivo tramite una connessione usb. La presenza del modulo GPS, sincronizzato con i satelliti, oltre a fornire il timestamp di ogni acquisizione, da anche la capacità di sincronizzare le schede di tutti gli apparati, ovunque essi si trovino. In questo modo è possibile far patire le acquisizioni dei vari siti di monitoraggio tutte nello stesso istante temporale.

In Figura 3.6 è mostrata un'immagine della scheda di acquisizione appena descritta e del suo case, si possono distinguere i blocchi principali: l'ADC (evidenziato in blu), la CPLD (in giallo), il modulo GPS (in rosso), i partitori di tensione (in viola) ed il convertitore seriale/usb (in arancione).

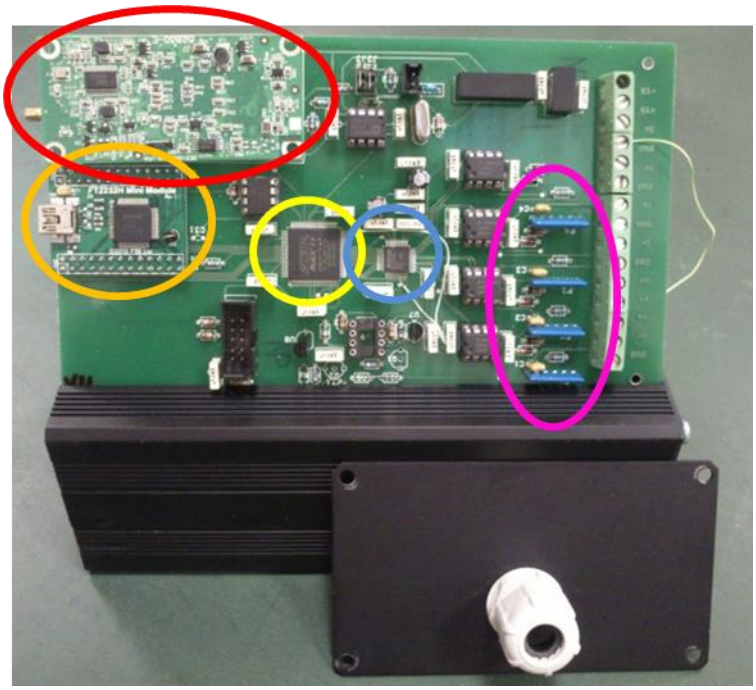


Figura 3.6 - Fotografia della scheda di acquisizione con in evidenza i componenti principali.

3.1.2 Il micro-computer o uPC

Il dispositivo scelto per sostituire il desktop pc del vecchio sistema è il BeagleBone Black, un micro-computer a basso costo realizzato inizialmente dalla società CircuitCo e poi anche da element14, in vendita ad un prezzo di circa 50 $. Le caratteristiche ufficiali dichiarate di questa scheda elettronica basata sul processore prodotto da Texas Instruments (TI) Sitara AM3358 sono mostrate in Tabella 3.1.

	Feature	
Processor	Sitara AM3358BZCZ100 1GHz, 2000 MIPS	
Graphics Engine	SGX530 3D, 20M Polygons/S	
SDRAM Memory	512MB DDR3L 800MHZ	
Onboard Flash	4GB, 8bit Embedded MMC	
PMIC	TPS65217C PMIC regulator and one additional LDO.	
Debug Support	Optional Onboard 20-pin CTI JTAG, Serial Header	
Power Source	miniUSB USB or DC Jack	5VDC External Via Expansion Header
PCB	3.4" x 2.1"	6 layers
Indicators	1-Power, 2-Ethernet, 4-User Controllable LEDs	
HS USB 2.0 Client Port	Access to USB0, Client mode via miniUSB	
HS USB 2.0 Host Port	Access to USB1, Type A Socket, 500mA LS/FS/HS	
Serial Port	UART0 access via 6 pin 3.3V TTL Header. Header is populated	
Ethernet	10/100, RJ45	
SD/MMC Connector	microSD , 3.3V	
User Input	Reset Button Boot Button Power Button	
Video Out	16b HDMI, 1280x1024 (MAX) 1024x768,1280x720,1440x900 ,1920x1080@24Hz w/EDID Support	
Audio	Via HDMI Interface, Stereo	
Expansion Connectors	Power 5V, 3.3V , VDD_ADC(1.8V) 3.3V I/O on all signals McASP0, SPI1, I2C, GPIO(69 max), LCD, GPMC, MMC1, MMC2, 7 AIN(1.8V MAX), 4 Timers, 4 Serial Ports, CAN0, EHRPWM(0,2),XDMA Interrupt, Power button, Expansion Board ID (Up to 4 can be stacked)	
Weight	1.4 oz (39.68 grams)	
Power	Refer to Section 6.1.7	

Tabella 3.1 - Caratteristiche del BeagleBone Black (fonte: elinux.org).

Da cui si possono estrarre i punti chiave utili al nostro progetto:

- Processore ARM Cortex-A8 da 1 GHz prodotto da TI, con elevata capacità di calcolo ed alta affidabilità;
- Memoria RAM da 512 MB ad 800 MHz;
- Memoria Flash on-board da 4 GB, con la possibilità di aggiungere ulteriore memoria di massa utilizzando schede SD;
- Varie porte di connessione come Ethernet, USB, HDMI e varie porte seriali;
- Fino a 69 General Purpose Input/Output (GPIO);
- Dimensioni della scheda molto ridotte;

- Consumi molto bassi che vanno da 210 mA a 460 mA ad una tensione di 5 V (senza dispositivi usb collegati);
- Possibilità di installare un sistema operativo, di base viene fornito con la distribuzione Linux di Angstrom o di Debian preinstallata nella flash, ma questa può facilmente essere sostituita con altri sistemi operativi compilati per processori Arm.

Quindi è praticamente un computer a tutti gli effetti, dotato di S.O., con consumi davvero ridotti, a basso costo e nelle dimensioni contenute di una carta di credito.

Essendo "open-source hardware" si possono trovare facilmente tutti gli schematici di questo dispositivo [BeagleBone], che permettono di avere una conoscenza completa dei suoi componenti e delle sue funzionalità, dando grandi vantaggi in fase di sviluppo.

Inoltre, questa piattaforma ha dalla sua parte la possibilità di essere usata subito "così com'è" semplicemente collegandola ad un computer con un cavo usb e cominciando a programmare. Questo è stato molto utile in fase iniziale di "avvicinamento" al device, poi però, è stato necessario un radicale cambiamento del sistema, come verrà mostrato e spiegato più avanti nel paragrafo dedicato al software del BeagleBone Black.

In Figura 3.7 è mostrata un'immagine reale del uPC in questione.

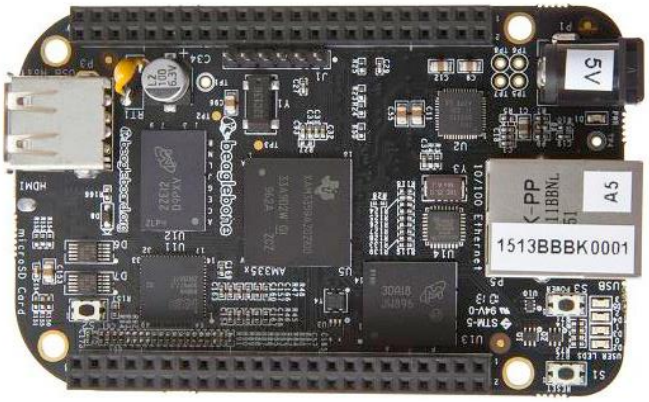


Figura 3.7 - Immagine del BeagleBone Black (fonte: BeagleBoard.org).

3.1.3 Il controller telefonico

Nel sistema è previsto un controller telefonico che in caso di problemi e in assenza di comunicazione tramite la linea dati, permette, tramite una semplice telefonata, di

fornire informazioni sullo stato del uPC, lo stato della batteria ed, eventualmente, è in grado di riavviare il sistema.

Questo dispositivo, realizzato dal laboratorio di Misure nel corso degli anni, può essere schematizzato come in Figura 3.8.

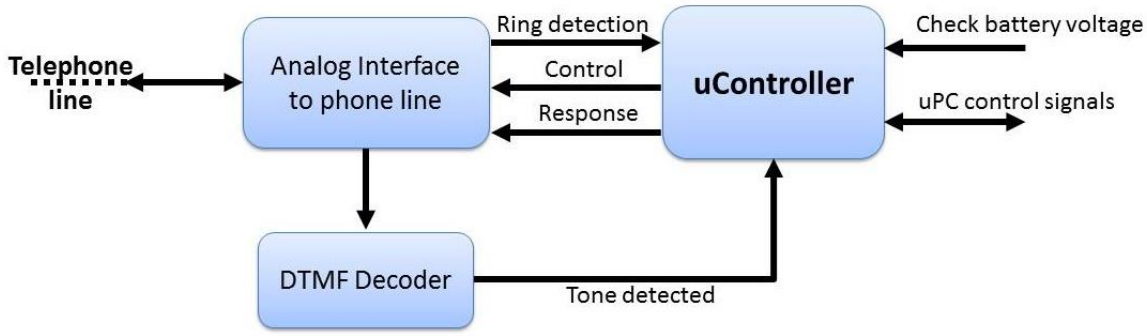


Figura 3.8 - Schema di principio del controller telefonico.

Il primo blocco analogico, costituito da filtri, un trasformatore di linea ed un relè per simulare l'alzata della cornetta del telefono, permette di interfacciare il controller con la linea telefonica.

Quando il dispositivo viene chiamato, da un qualunque apparecchio telefonico, il microcontrollore rileva la presenza degli squilli grazie al segnale di "ring detection" ed al quarto squillo risponde pilotando l'interfaccia analogica in modo che simuli l'alzata della cornetta. A questo punto il chiamante, riconoscendo che il sistema ha risposto, può comunicare con il microcontrollore digitando i numeri dalla tastiera del telefono. Il uC, grazie all'ausilio del decoder di toni DTMF, riesce a riconoscere i tasti premuti dall'utente e quindi può eseguire le operazioni richieste e comunicarne l'esito (positivo o negativo) all'interlocutore, tramite la generazione di 2 toni a frequenze differenti. Queste frequenze vengono generate da due treni di impulsi che tramite il segnale di "response" vengono filtrati dall'interfaccia analogica e raggiungono l'altro capo della comunicazione.

In questo modo, il microcontrollore, può essere programmato per eseguire fino a 12 operazioni, tante quanti sono i tasti di un comune telefono. Attualmente, a seconda del tasto premuto, vengono eseguite le operazioni di:

- check della tensione di batteria, se troppo bassa viene emesso un tono basso a 470 Hz, altrimenti un tono alto a 1100 Hz;
- check dello stato del uPC, (acceso=tono alto, spento=tono basso);

- riavvio del uPC, inviando un comando di reset allo stesso;
- chiusura della comunicazione telefonica, riaprendo il relè e simulando l'abbassamento della cornetta.

Inoltre, qualora la connessione non venisse chiusa volontariamente premendo il tasto corrispondente, il microcontrollore provvede a terminare la chiamata automaticamente dopo un periodo di inattività dell'utente di 60 s.

3.1.4 Il modem ADSL

La presenza di una connessione dati è fondamentale per il buon funzionamento dell'apparecchiatura di monitoraggio realizzata, in quanto:

- Fornisce la possibilità di trasferire automaticamente i dati acquisiti su un server centrale, liberando memoria nel sistema locale e permettendo di concentrare i dati di monitoraggio per ulteriori elaborazioni;
- Permette l'accesso da remoto al uPC, per effettuare, quando necessario, operazioni di manutenzione e di aggiornamento software.

Il primo punto verrà trattato più nel dettaglio nel paragrafo riguardante il software che gira sul BeagleBone, mentre il secondo punto verrà spiegato nel paragrafo inerente la Gestione e il Controllo dell'infrastruttura.

Essendo i siti di monitoraggio del MeaLab installati all'interno di centrali di Telecom Italia, si ha a disposizione una linea ADSL, con modem/router ethernet dedicato, per ogni sito. I modem in questione, tra le varie opzioni di configurazione, hanno anche il "**port forwarding**", questo permette di aprire delle porte specifiche del router e di ri-dirigere il flusso di dati diretto verso queste porte e proveniente dall'esterno verso uno specifico dispositivo connesso nella lan interna al router. Quindi, questi modem sono stati configurati in modo tale che quando un qualsiasi utente di internet, da un qualunque dispositivo connesso in rete, tenta di collegarsi a queste porte, tutto il traffico viene dirottato sul BeagleBone Black connesso sulla rete interna del router. In questa maniera si può accedere al uPC direttamente come se fosse collegato sulla stessa lan privata dell'operatore. In Figura 3.9 è mostrata una schermata della configurazione del router che permette questo re-indirizzamento. Le porte che vengono utilizzate, come si può vedere in figura, sono la 22 per l'SSH e la 10000 per Webmin, questi servizi per la gestione del sistema verranno mostrati più avanti.

Figura 3.9 - Interfaccia del modem/router per configurare il port forwarding.

Nel caso non sia possibile una connessione ADSL come appena descritta (modem/router dedicato al probe e con port-forwarding) è possibile utilizzare anche altri tipi di collegamento. Sono state testate le seguenti ulteriori modalità di connessione:

- Ethernet ADSL modem/router senza port-forwarding;
- Wi-Fi ADSL modem/router con e senza port-forwarding, utilizzando una chiavetta USB/Wi-Fi per connettere il uPC alla rete wireless;
- Wi-Fi 3G modem/router senza port-forwarding, dotando anche in questo caso il BeagleBone Black di una scheda di rete USB/Wi-Fi.

Senza port-forwarding non c'è nessun tipo di problema per la trasmissione dei file al server, ma non è possibile accedere direttamente al sistema da remoto per fare manutenzione. Per ovviare a questo inconveniente è stata studiata una soluzione che permette comunque di manutenere i sistemi da remoto, facendo "***Tunneling***" attraverso il server. Questo metodo di collegamento, essendo tutto strutturato a livello software, verrà trattato nei paragrafi riguardanti la gestione del sistema.

3.1.5 Il sistema di alimentazione continuo

Per garantire che la strumentazione rimanga sempre attiva, anche durante le interruzioni della rete di distribuzione, in modo da poter monitorare i transitori e i disturbi che si verificano al ritorno dell'energia elettrica, è stato implementato un

sistema di alimentazione continuo. Questo sistema è costituito da un alimentatore stabilizzato di tipo switching (Figura 3.10) e da una batteria in tampone costantemente collegata (Figura 3.11).

Figura 3.10 - Alimentatore della TDK-Lambda scelto per il progetto (fonte: RS Components).

Figura 3.11 - Batteria prodotta da RS ed utilizzata nel sistema (fonte: RS Components).

Il consumo energetico massimo dell'intero punto di monitoraggio è stato misurato sperimentalmente e si attesta intorno ai 5W. Una delle specifiche dell'intero sistema è quella di avere un'autonomia energetica minima di 36 ore in caso di black out, perciò è stato stimato che con una batteria a 12V da 12Ah si riesce tranquillamente a soddisfare questo requisito (prove sperimentali hanno evidenziato un'autonomia del sistema superiore alle 48h).

Come batteria tampone, quindi, è stato scelto un accumulatore al piombo ricaricabile di tipo sigillato senza manutenzione da 12V e 12Ah, prodotto da RS Components e dal costo di circa 25€.

L'alimentatore è stato scelto in base ai seguenti criteri:

- Regolazione della tensione d'uscita per adattarla a quella necessaria per mantenere la batteria in tampone senza rovinarla, che ad una temperatura ambiente di 25°C deve essere di 13.5÷13.8V;

- Corrente d'uscita in grado di fornire contemporaneamente sia la corrente necessaria ad alimentare tutta la strumentazione sia la corrente massima di carica prevista per la batteria collegata, senza surriscaldamento dell'alimentatore e senza diminuzione della tensione;
- Dimensioni contenute, per rientrare nell'obiettivo di ridurre gli ingombri del punto di monitoraggio;
- Peso limitato, optando per un modello switching si riducono notevolmente i pesi a parità di potenza erogata;
- Basso costo, sempre per rientrare negli obiettivi globali.

Seguendo questi requisiti, è stato selezionato un alimentatore da 15V prodotto dalla società TDK-Lambda con le seguenti caratteristiche peculiari:

- Tensione di uscita regolabile nel range 13.5÷16.5V;
- Corrente d'uscita massima di 5A;
- Dimensioni e peso contenuti (130 mm x 97 mm x 38 mm e 410 g);
- Elevata efficienza, tipica dell'85%;
- Costo intorno ai 30€;
- Protezione contro sovratensioni e sovracorrenti;
- Ingresso universale: AC 88÷264V a 47÷63Hz, DC 125÷373V.

Infine, per portare la tensione a 5V necessaria per il funzionamento del BeagleBone Black, si è utilizzato un convertitore DC/DC switching ad elevata efficienza (maggiore del 92%) prodotto da Lineage Power, in grado di fornire tutta l'energia necessaria al uPC ed ai dispositivi collegati ad esso. Il costo di questo dispositivo (mostrato in Figura 3.12) si aggira intorno ai 5€, ha peso trascurabile e dimensioni inferiori a quelle di un francobollo (12.2 mm x 12.2 mm x 6.25mm).

Figura 3.12 - Il convertitore DC/DC utilizzato per ridurre la tensione di batteria a 5V (fonte: RS Components).

3.1.6 Il server

Come si è già detto, il uPC una volta elaborati i dati ricevuti dalla scheda di acquisizione li trasmette al server centrale, in questo modo possono essere centralizzati tutti i dati dei vari punti di monitoraggio, per poter essere visualizzati da delle opportune interfacce web, oppure per essere ulteriormente elaborati. Ciò rende il server un componente indispensabile all'interno dell'infrastruttura. Inoltre, la sua presenza è fondamentale anche per il controllo dei vari siti di monitoraggio. Questi ultimi due punti saranno molto più chiari nei successivi paragrafi, quando verrà mostrata la parte software e quella di gestione dell'infrastruttura.

Dal punto di vista dell'hardware non è richiesto un qualcosa di particolare, purchè sia abbastanza performante da gestire tutte le connessioni con i vari siti di monitoraggio senza problemi e sia in grado di far girare il software mostrato nei paragrafi seguenti in maniera fluida.

Per completezza viene data comunque qualche informazione sulle caratteristiche hardware del server utilizzato dal MeaLab in questo progetto. Il sistema in questione è quindi dotato di processore Intel Core i7 (quad-core a 3GHz) equipaggiato con 8GB di RAM DDR3 e 1TB di hard-disk. Inoltre è sempre connesso alla rete internet tramite una connessione a 100Mbit ed è alimentato tramite un sistema UPS (Uninterruptible Power Supply).

3.2 Il Software

In questa sezione verrà descritto il software realizzato ed installato sui dispositivi, analizzando prima quello residente sui BeagleBone Black dei vari strumenti di monitoraggio e poi quello implementato nel server.

3.2.1 Lato BeagleBone Black

La descrizione del software che gira sul uPC parlerà sia del Sistema Operativo, sia dei vari servizi accessori, sia degli applicativi sviluppati che sono quelli che si occupano del monitoraggio vero e proprio.

Nella maggior parte dei casi, quando si sviluppa un software per computer, non si da troppa importanza al S.O., si sta giusto a vedere di che tipo è (Windows, Linux o Mac OS) e la versione. Questo perchè spesso o ci si trova ad utilizzarne uno preinstallato nella macchina o comunque, se se ne installa uno appositamente, non lo si va a modificare troppo nei suoi servizi, in quanto, solitamente ciò non comporterebbe molta differenza a livello di prestazioni del software.

In questo caso, invece, dato l'hardware particolare (un uPC) e l'utilizzo specifico (un sistema remoto), due delle operazioni più importanti, preliminari alla stesura del codice che si occupa del monitoraggio, sono state proprio la scelta del Sistema Operativo e la sua "configurazione" con tutti i servizi necessari. Questo ha permesso di avere una solida piattaforma software su cui potercisi appoggiare per poi sviluppare e testare il codice per la Power Quality.

Nei seguenti paragrafi verrà prima descritta la scelta del S.O. e dei servizi installati ritenuti importanti e poi verranno mostrati gli applicativi per il monitoraggio sviluppati.

3.2.1.1 Sistema Operativo ed alcuni servizi

Il BeagleBone Black è una piattaforma di sviluppo open-hardware su cui possono essere installati numerosi Sistemi Operativi, purchè, ovviamente, questi siano stati compilati per architettura Arm e siano compatibili con l'hardware in questione.

All'acquisto viene fornito già con un S.O. preinstallato, ma se ne possono installare anche altri, quali ad esempio:

- Android;
- Varie distribuzioni Linux come: Angstrom, Debian, Ubuntu, Fedora, ArchLinux, Buildroot, Gentoo, Yocto, ecc;
- MINIX, FreeBSD (S.O. liberi di tipo Unix);
- XNU (kernel alla base anche di Mac OS X);
- Windows Embedded Compact (Windows CE).

Di questi Sistemi Operativi ne sono disponibili anche diverse versioni.

Quando si è cominciato a lavorare su questo progetto nel 2013, il BeagleBone Black era da poco uscito in commercio ed era venduto con una memoria flash da 2GB su cui era preinstallata la distribuzione Linux di Angstrom. Dopo una prima fase di avvicinamento a questo nuovo S.O., ci si accorse che era molto minimale, forse troppo e infatti si sentiva la mancanza di qualche servizio in più che sarebbe potuto risultare utile nello sviluppo del sistema. Inoltre, era anche poco "user-friendly". Di conseguenza si decise, vista la possibilità, di cambiare Sistema Operativo, scegliendone uno più adatto al nostro caso.

La selezione del nuovo S.O. da caricare nel BeagleBone Black è stata guidata dai seguenti criteri e necessità:

- Doveva occupare poco spazio sulla memoria flash, in modo da lasciare più spazio per i file acquisiti durante il monitoraggio;
- Doveva essere leggero dal punto di vista computazionale, in modo da non appesantire il sistema e renderlo "reattivo", lasciando molte risorse libere per il calcolo degli algoritmi di Power Quality. Inoltre, non impegnando troppo la CPU se ne avrebbe guadagnato anche in termini di consumi energetici;
- Doveva essere sviluppato ed ottimizzato per l'hardware dalle prestazioni abbastanza limitate del BeagleBone Black;
- Doveva essere facilmente configurabile, anche per aggiungere o togliere moduli necessari o non;
- Doveva essere facilmente utilizzabile;
- Doveva avere un costo minore possibile.

Con tutti questi criteri la scelta è rimasta su Linux, ma si è spostata sulla distribuzione di Ubuntu (Versione 12.04 LTS - Long Term Support - poi aggiornata alla 14.04 LTS) anche per una certa familiarità con questo S.O. e per la vasta community alle sue spalle, in grado di fornire informazioni e supporto in caso di bisogno.

Ubuntu è una distribuzione GNU/Linux basata su Debian, che ha la caratteristica di essere molto "user-friendly" e quindi facilmente configurabile. Inoltre, viene pubblicata come software libero sotto licenza GNU GPL (General Public License), è distribuita gratuitamente ed è liberamente modificabile.

Di questo S.O. si trovano liberamente su internet diverse versioni compilate ed adatte a girare sul BeagleBone Black, ognuna con moduli o servizi differenti che la rendono più o meno pesante a livello di occupazione di memoria e di altre risorse.

Inizialmente si pensava di poter utilizzare direttamente una di queste versioni, così, dopo una lunga ricerca di quali potevano essere quelle più adatte al nostro progetto, si è passati a testarle una ad una, installandole sul dispositivo ed analizzando:

- quali servizi erano in esecuzione;
- la quantità di memoria flash occupata;
- l'utilizzo medio della CPU;
- l'occupazione media della memoria RAM.

Da questi test non si è però riusciti a trovare un S.O. che fosse "già pronto" da poter essere utilizzato direttamente nel nostro progetto, in quanto le versioni di Ubuntu risultavano o troppo avide di risorse hardware (spesso a causa dell'interfaccia grafica e di varie applicazioni) o troppo minimaliste (e quindi carenti di alcune funzionalità).

A questo punto si è pensato di provare a "configurare" ad hoc una di queste versioni di Ubuntu, o meglio, si sono percorse due strade parallelamente: una che prevedeva di "alleggerire" una delle versioni "pesanti" ed un'altra strada che, invece, prevedeva di "appesantire" una delle versioni "leggere".

Essendo l'interfaccia grafica uno dei componenti del S.O. che utilizzano maggiori risorse, si è deciso di utilizzare nel progetto una versione di Ubuntu senza questa estensione e quindi solo con interfaccia testuale a linea di comando (CLI – Command Line Interface). Ciò permette di liberare molte risorse hardware a fronte di una interfaccia meno user-friendly, ma dato che è previsto che i punti di monitoraggio siano autonomi e che bisogna intervenire manualmente solo per manutenzione, si è deciso che questo non fosse un grave problema. Inoltre, dato che gli apparati del progetto sono pensati per essere "headless", quindi senza monitor, tastiera ne mouse e gestiti completamente da remoto, il non avere un'interfaccia grafica rende anche la connessione da remoto molto più leggera.

Quindi, per la prima strada, la prima operazione per "alleggerire" Ubuntu è stata quella di eliminare la GUI (Graphical User Interface) rimuovendo i vari pacchetti dal S.O..

Poi sono state eliminate anche tutte quelle applicazioni che richiedevano l'uso della grafica, perché oramai erano inutilizzabili. Questa operazione si è rilevata abbastanza lenta e laboriosa, in quanto prima di usare il comando di rimozione:

apt-get purge --auto-remove "nome_pacchetto_da_rimuovere"

è stato necessario andare a rovistare tra i centinaia di pacchetti installati in Ubuntu, per cercare di capire quali erano legati alla GUI e quali non servivano (andando continuamente su internet, da un altro PC, per trovare informazioni su questi pacchetti). Il comando utilizzato per listare tutti i "packages" installati è:

dpkg -l

Inoltre, tutto il lavoro è stato svolto in un ambiente completamente testuale, quindi poco "confortevole", e con la necessità spesso di riavviare il uPC, aggiornarlo ed andare a ricontrollare nuovamente i pacchetti.

Alla fine di questa lenta procedura, si è arrivati ad un sistema senza GUI, abbastanza leggero e poco ingombrante sulla flash (anche se erano ancora da installare diversi servizi necessari), infatti per l'utilizzo medio della RAM si è passati da circa 280 MB a circa 50 MB, mentre per la memoria di massa da circa 1.4 GB a circa 800 MB. Anche per quanto riguarda l'uso della CPU si aveva una percentuale molto bassa, ma ci si è accorti che si avevano in esecuzione o in background alcuni processi che non dovevano esserci, forse a causa della rimozione parziale di qualche dipendenza dei pacchetti eliminati. Questo ci ha portato a preferire la seconda strada, in cui molto probabilmente si avrebbe avuto un'istallazione "più pulita."

Nella seconda strada si è partiti da una versione del S.O. già leggera, infatti era già senza interfaccia grafica ed aveva solo la CLI e si è cercato di aggiungere le funzionalità mancanti. Dopo averla installata nella flash del uPC ed averla aggiornata si è passati a configurare alcune impostazioni e poi ad installare alcune funzioni.

In Appendice viene descritta la procedura utilizzata per preparare il BeagleBone Black, con tutti i passaggi da seguire spiegati più in dettaglio. Sommariamente questi passaggi possono essere così riassunti:

- Download su PC dell'immagine di Ubuntu per BeagleBone Black;
- Preparazione di una memoria uSD da inserire nel uPC;
- Scrittura dell'immagine di Ubuntu sulla eMMC;

- Configurazione della connessione internet, a seconda della rete LAN a cui ci si deve connettere;
- Aggiornamento del S.O., eventualmente anche della distribuzione Linux;
- Impostazione lingua e layout tastiera;
- Installazione Webmin, è un'interfaccia web per amministratori di sistema e permette di collegarsi al sistema da remoto tramite un'interfaccia grafica su browser (essendo utilizzata per la gestione del uPC se ne parlerà più avanti nel rispettivo paragrafo);
- Controllo degli utenti con accesso al sistema, per essere sicuri che *di default* non ci siano altri utenti, oltre a quello principale, in grado di accedere al sistema da remoto;
- Installazione servizio NTP e server I.N.Ri.M., è un servizio che permette di mantenere l'orologio di sistema costantemente aggiornato con i server NTP (Network Time Protocol), correggendone continuamente la deriva. Il demone NTP è stato configurato per utilizzare, oltre ai server di base, anche quelli dell'Istituto Nazionale di Ricerca Metrologica;
- Modifica della password di accesso, che altrimenti sarebbe quella di default per tutte le installazioni e che è pubblicata sul sito dove si è scaricata l'immagine di Ubuntu;
- Installazione firewall e apertura porte, l'apertura delle porte è necessaria per permettere l'accesso al sistema per la gestione da remoto (sono, ovviamente, le stesse porte che vanno aperte sul modem/router, paragrafo 3.1.4);
- Cancellazione di user e password nella schermata di login, altrimenti sarebbero visibili a chiunque *prima* di effettuare il login;
- Riavvio e ri-aggiornamento del sistema, necessario anche per vedere se ci sono stati problemi durante qualche installazione.

La "configurazione" ad hoc del sistema ha, ovviamente, aumentato leggermente l'utilizzo di risorse hardware, rispetto alla versione "non configurata" (soprattutto a causa dell'installazione di Webmin), però queste si sono comunque mantenute basse. Per quanto riguarda l'utilizzo medio della RAM si è passati da circa 35 MB a circa 90 MB, per la flash da circa 400 MB a circa 700 MB, invece la CPU ha un carico medio su 15 minuti inferiore all'1%.

Alla fine, quindi, si è ottenuta una versione di Ubuntu aggiornata, reattiva, leggera e con tutti i servizi necessari e utili per il nostro progetto:

- No GUI, per alleggerire il sistema e perché comunque non sarebbe visibile da remoto;
- NTP, per mantenere sincronizzato l'orologio di sistema. Molto importante anche per mantenere sincronizzate tutte le operazioni pianificate;
- Firewall, per cercare di evitare attacchi da malintenzionati, dato che il sistema deve essere accessibile dall'esterno per la manutenzione;
- SSH server, di base nei sistemi Linux, per il controllo, l'aggiornamento e la manutenzione del sistema embedded da linea di comando;
- Webmin, per la gestione "grafica" da remoto (se necessaria);
- FTP-Curl, per permettere al software di monitoraggio il trasferimento dei file sul server centrale (mostrato nel prossimo paragrafo).

I servizi SSH e Webmin, verranno illustrati nel paragrafo dedicato alla gestione ed al controllo dell'intera infrastruttura.

3.2.1.2 Applicativi sviluppati per il monitoraggio

Una volta definita la "piattaforma software" da utilizzare sul BeagleBone Black, si è passati allo sviluppo delle applicazioni che si sarebbero occupate del monitoraggio vero e proprio.

Date le limitate risorse hardware si è pensato di utilizzare un linguaggio di programmazione non troppo di alto livello ma più vicino all'hardware, in modo da poter progettare i vari processi applicativi con più attenzione sulle risorse utilizzate. Quindi, per sviluppare questi software applicativi si è deciso di utilizzare il linguaggio C e data la scelta di Linux come Sistema Operativo, si è utilizzato il suo compilatore nativo GCC (ora "GNU Compiler Collection", ma in origine "GNU C Compiler"). Questo compilatore, nato inizialmente per il linguaggio C ma ora in grado di supportarne anche altri come C, C++, Objective-C, Fortran, Java, Ada e Go, è anche open source e si avvale di una licenza libera GNU GPL. Per poter sviluppare il

software, perciò, si è installato GCC e le sue librerie su uno dei BeagleBone Black preparati con il S.O. descritto al paragrafo precedente. In questo modo, il software veniva compilato sempre sullo stesso uPC in laboratorio, per poi distribuire, sugli altri sistemi embedded, soltanto i file compilati eseguibili, evitando di appesantire inutilmente i sistemi con il compilatore e le sue librerie.

Il software per il monitoraggio che si è sviluppato, fondamentalmente, si occupa di acquisire i dati ed il Timestamp dalla scheda di acquisizione, di elaborarli e di inviare al server centrale i file così generati. In Figura 3.13 è mostrato il principio di funzionamento del software.

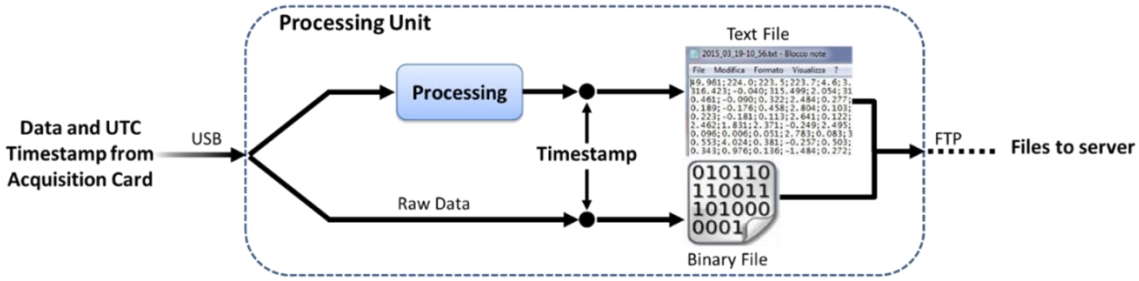


Figura 3.13 - Principio di funzionamento del software di monitoraggio sul uPC.

Per assolvere i suoi compiti appieno, il software, è stato strutturato in 3 programmi separati, ognuno con compiti diversi:

- Il primo, "***configuration***", si occupa della preparazione del sistema;
- Il secondo, "***capture***", si occupa dell'acquisizione dei dati, della loro elaborazione e della generazione dei file elaborati;
- Il terzo, "***ftp_sender***", gestisce l'invio dei file al server.

Verranno adesso analizzati singolarmente per mostrare il loro principio di funzionamento, per un'analisi più approfondita si rimanda al codice completo allegato in Appendice.

3.2.1.2.1 "configuration"

Questo eseguibile serve per la configurazione della macchina locale e va lanciato solo all'inizio dopo aver messo a terra tutti gli ingressi analogici.

Fornisce *un'identità* alla macchina, necessaria anche al server per riconoscerla quando gli trasmette i dati con l'altro applicativo ed inoltre permette di misurare e registrare l'offset della scheda di acquisizione, per la successiva compensazione eseguita dal

programma di elaborazione. Crea anche un file di configurazione con questi parametri, che gli altri programmi andranno a leggere.

In Figura 3.14 viene mostrato un diagramma di flusso funzionale di questo applicativo.

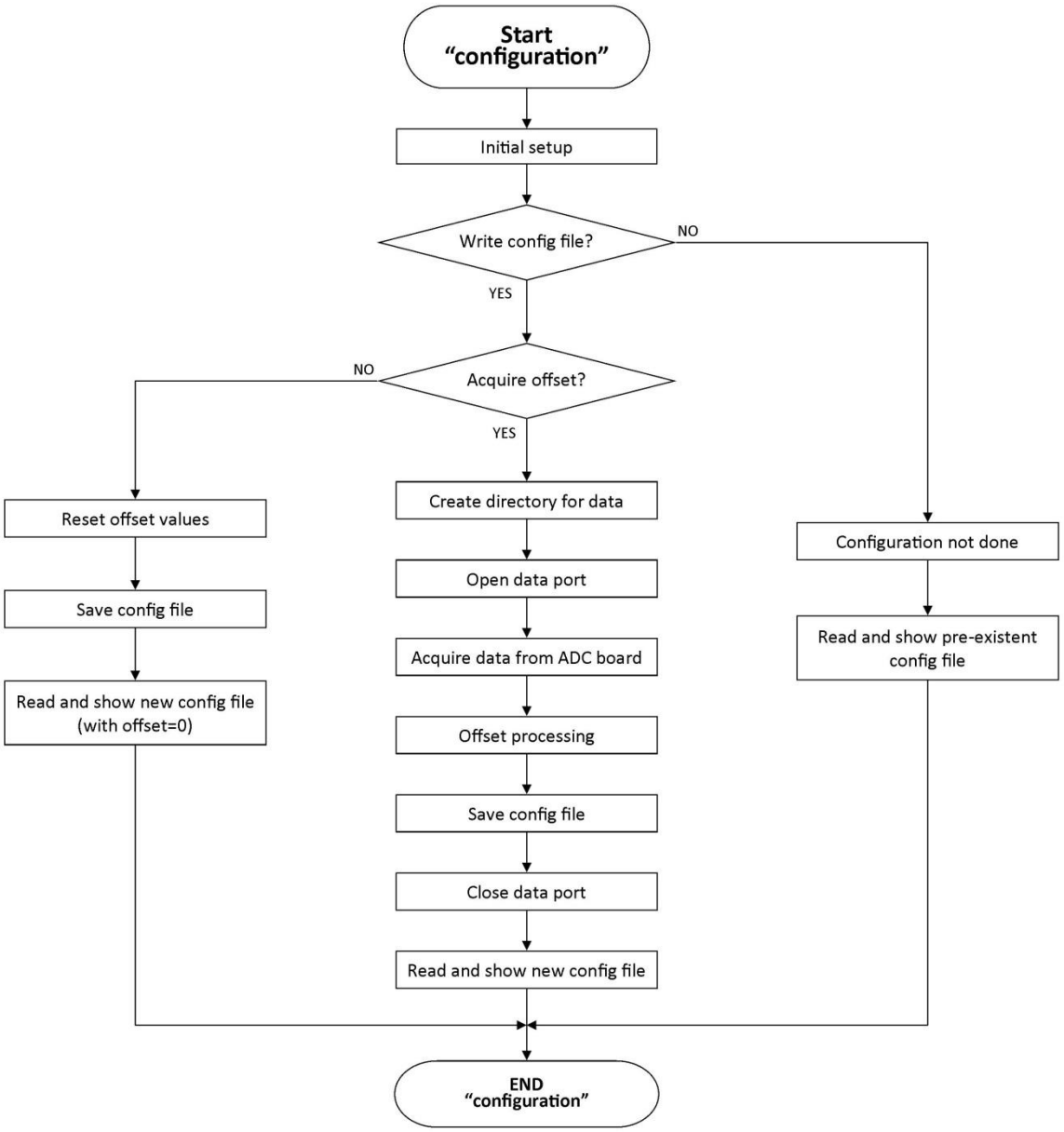


Figura 3.14 – Flow chart di "configuration".

Una volta lanciato, dopo una prima fase iniziale di setup dell'applicativo, viene chiesto all'utente se vuole scrivere/riscrivere il file di configurazione e se vuole scriverlo misurando gli offset della scheda oppure no. Questo perché per misurare gli offset di ogni canale è necessario che gli ingressi analogici vengano collegati tutti a terra (che è il potenziale di riferimento della scheda), ma questa operazione potrebbe momentaneamente non essere possibile, in questo caso la generazione di un file di

configurazione con gli offset a zero permetterebbe comunque alla macchina di avere un'identità e agli altri 2 applicativi di poter lavorare senza problemi. Successivamente si potrebbe poi sempre rilanciare il "configuration" per riscrivere il file con gli offset calcolati.

Il diagramma di flusso risulta abbastanza esplicativo per capire quali sono i passaggi che vengono eseguiti una volta scelto uno dei tre percorsi.

La scheda di acquisizione, come si è già detto, comunica con il uPC tramite un convertitore seriale/USB, quindi il collegamento USB viene visto da Linux come 2 porte seriali virtuali e gli applicativi, quindi, acquisiscono i dati dalla scheda sfruttando le librerie di Linux per la gestione (molto a basso livello) di porte seriali.

Ogni campione è a 14 bit e viene perciò spezzato in 2 byte, di conseguenza, i dati provenienti ogni minuto dall'ADC equivalgono a:

$$1500\ acquisizioni \quad x \quad 8\ canali \quad x \quad 2\ byte \quad = \quad 24000\ byte$$

Sperimentalmente si è visto che questi 24000B arrivano spezzettati in vari pacchetti ed a volte può succedere che qualche pacchetto si perda o arrivi in ritardo con il "burst" successivo. Per ovviare a questo si è messo in piedi, a livello software, un controllo per verificare che i byte ricevuti siano esattamente 24000 ogni minuto, ne più ne meno, e che arrivino tutti entro un certo tempo. In caso di anomalie, è previsto che la porta seriale in questione venga resettata e che venga svuotato sia il suo buffer fisico che quello software, in modo da essere "pulita" e pronta a ricevere i dati per il minuto successivo.

Avendo a disposizione 1500 campioni per ogni canale, per calcolare l'offset viene fatta la media di tutte queste acquisizioni (operazione di "offset processing" vista nel diagramma di flusso e mostrata nel dettaglio in Figura 3.15).

```
    // Fa la media dei valori acquisiti sulle singole fasi e la carica come offset
    for(i=0; i<=1499; i++) {
        accVr += Vr[i];
        accVs += Vs[i];
        accVt += Vt[i];
        accVn += Vn[i];
    }
    offsetVR = (accVr / 1500);
    offsetVS = (accVs / 1500);
    offsetVT = (accVt / 1500);
    offsetVN = (accVn / 1500);

}   // Fine offset_processing
```

Figura 3.15 - Parte di codice per il calcolo degli offset.

Infine, a prescindere da quale delle tre strade è stata scelta, viene visualizzato all'utente il contenuto del file di configurazione presente nel uPC, in modo da poter verificare il nome della macchina e se le variazioni sono state correttamente effettuate.

3.2.1.2.2 "capture"

Questo è il programma principale ed è continuamente in ascolto sulla porta della scheda di acquisizione per riceve i dati provenienti dall'ADC ed il timestamp dal modulo GPS. Arrivati i dati, li elabora con un algoritmo non ricorsivo ai minimi quadrati sviluppato dal laboratorio di Misure e genera 2 file: un file binario con i dati grezzi dell'ADC ed un file di testo con i valori stimati dall'algoritmo.

In Figura 3.16 è mostrato il principio di funzionamento di questo applicativo, di cui ora verranno spiegati più nel dettaglio i vari passi.

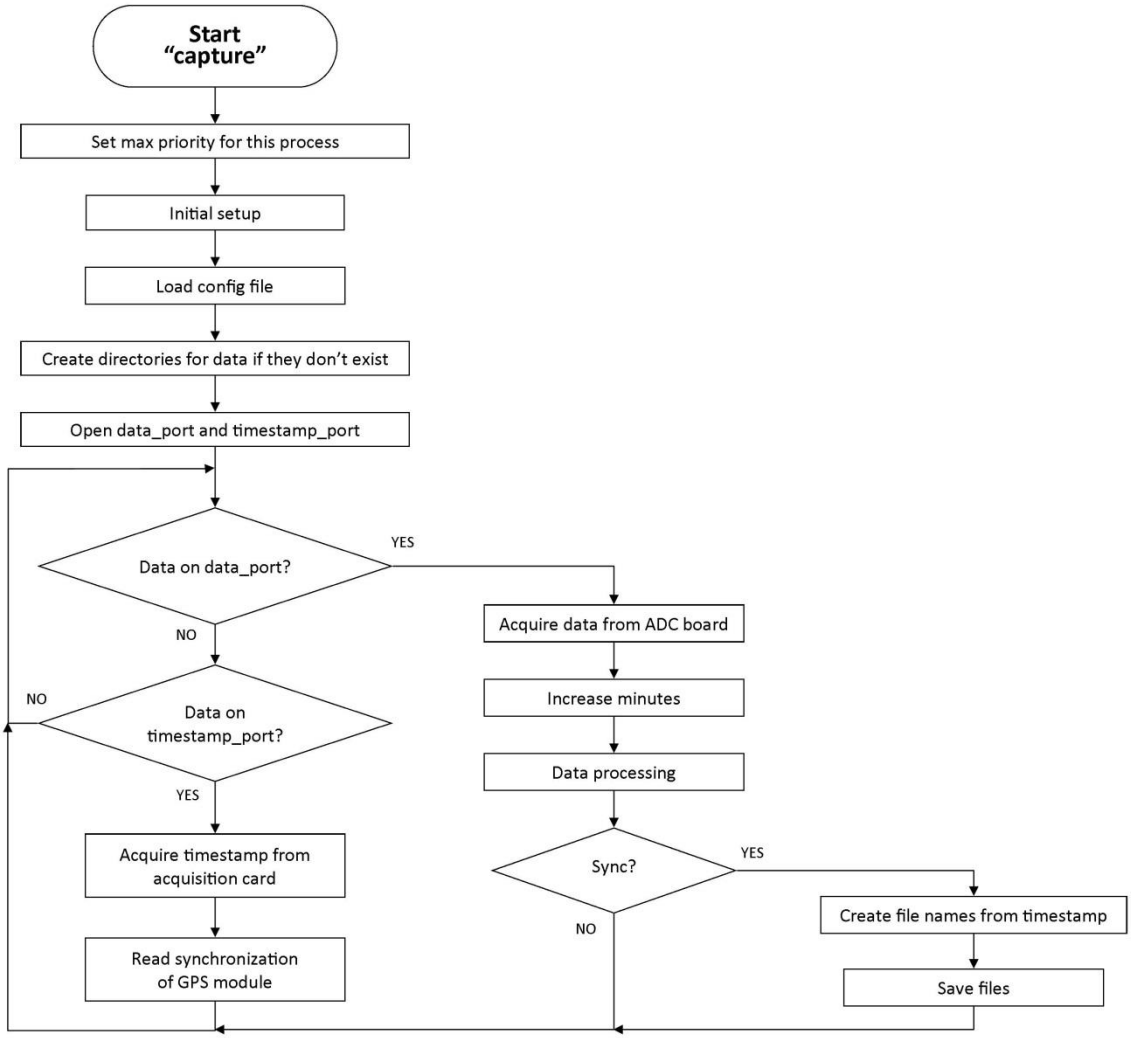


Figura 3.16 - Diagramma di flusso dell'applicativo "capture".

All’avvio, la prima operazione che viene eseguita è quella di impostare per questo processo la massima priorità di esecuzione, in modo che il sistema operativo lo gestisca come “prioritario” sull’utilizzo delle risorse.

Poi si ha una fase di setup iniziale, in cui viene anche caricato il contenuto del “config file” per la correzione degli offset, generato dal programma precedente, e vengono create le directory, se non presenti, dove verranno memorizzati i file del monitoraggio.

A questo punto, il sistema apre le porte di comunicazioni per ricevere i dati acquisiti dall’ADC ed il timestamp del modulo GPS e rimane continuamente in ascolto per la ricezione.

Dalla scheda arriveranno prima i 24000 byte e subito dopo il timestamp inerente quel burst di campioni. Oltre alla data e all’orario, dal modulo GPS, arriva anche un bit che ci dice se il modulo è sincronizzato o meno con i satelliti, questo per validare quel

timestamp, in quanto, se il modulo rimane non sincronizzato per un lungo periodo, il suo orologio interno continuerà a fornire un timestamp, ma questo sarà affetto dalla deriva di questo orologio che potrebbe essere anche notevole. Questo controllo è molto utile soprattutto appena acceso il sistema, quando il modulo non è sincronizzato con i satelliti e fornisce un timestamp casuale, in questo caso, infatti, come si può vedere dal diagramma, i campioni vengono ricevuti ed elaborati ma non salvati, dato che non si saprebbe a quale timestamp sono riferiti.

Essendo le varie acquisizioni scandite da un segnale metrologico ad elevatissima accuratezza, che tramite la CPLD va a definire anche il minuto tra un burst ed il successivo, si ha che anche questo minuto è definito molto accuratamente. Di conseguenza è stato implementato anche un "calendario" all'interno del programma, che viene incrementato di un minuto ogni volta che arriva un burst di 24000B. In questo modo, una volta ricevuto il primo timestamp sincronizzato, anche se il modulo GPS perdesse la sincronizzazione, il programma andrebbe comunque avanti nell'esecuzione delle sue funzioni con un timestamp accurato.

Per quanto riguarda l'elaborazione dei dati per la Power Quality, è stato implementato, nell'applicativo, un blocco di calcolo che applica ai campioni l'algoritmo sviluppato nel corso degli anni dal laboratorio di Misure Elettriche ed Elettroniche dell'Università degli Studi Roma Tre. Questo algoritmo si basa su un approccio non ricorsivo dell'analisi multi-armonica ai minimi quadrati e permette di stimare con elevata accuratezza la frequenza della tensione di rete, per poi calcolare ampiezza e fase delle varie armoniche. Rispetto ai metodi usati più comunemente, come FFT (Fast Fourier Transform) e zero-crossing, questo metodo permette di ottenere ottime stime anche per brevi osservazioni (alcuni periodi della fondamentale). Per un'analisi approfondita di questo approccio si rimanda a [Giarnetti] dove è stato largamente discusso.

Nel "capture", oltre al controllo sui pacchetti ricevuti ogni minuto (che devono essere 24000B ne più ne meno) implementato anche in "configuration", è stato inserito anche un controllo sul tempo trascorso tra un burst di 24000B ed il successivo. Se questo tempo è maggiore di un minuto vuol dire che si è persa un'acquisizione e quindi bisogna smettere di incrementare l'orologio software, interrompere il salvataggio dei file ed aspettare un nuovo timestamp "valido".

Se tutto va nel verso giusto, ogni minuto, i file generati durante l'esecuzione di questo programma sono 2:

- un file binario, contenente i dati grezzi provenienti dall'ADC, utili per ulteriori successive elaborazioni;
- un file di testo, contenente tutti i parametri stimati dall'algoritmo, quali:
    - frequenza di rete;
    - tensione efficace di ogni fase (R-S-T e N);
    - THD (Total Harmonic Distortion);
    - modulo e fase di ogni armonica per le tre fasi R-S-T.

3.2.1.2.3 "ftp_sender"

Il terzo file eseguibile, anch'esso in esecuzione continua come il precedente, ogni 5 minuti prova ad instaurare una connessione FTP (File Transfer Protocol) col server per spedirgli tutti i file bin e txt che si sono accumulati nel frattempo. Se riesce nel trasferimento, i file vengono cancellati da locale, altrimenti, in caso di problemi di connessione, vengono mantenuti in memoria e proverà a spedirli al tentativo successivo. Data la capacità della eMMC del BeagleBone Black e le dimensioni dei file generati, nel caso di anomalie sulla connessione, si ha un'autonomia di memorizzazione di circa 2 mesi e mezzo prima che la memoria flash sia satura.

Lo schema a blocchi mostrato in Figura 3.17 rappresenta il funzionamento di questo applicativo.

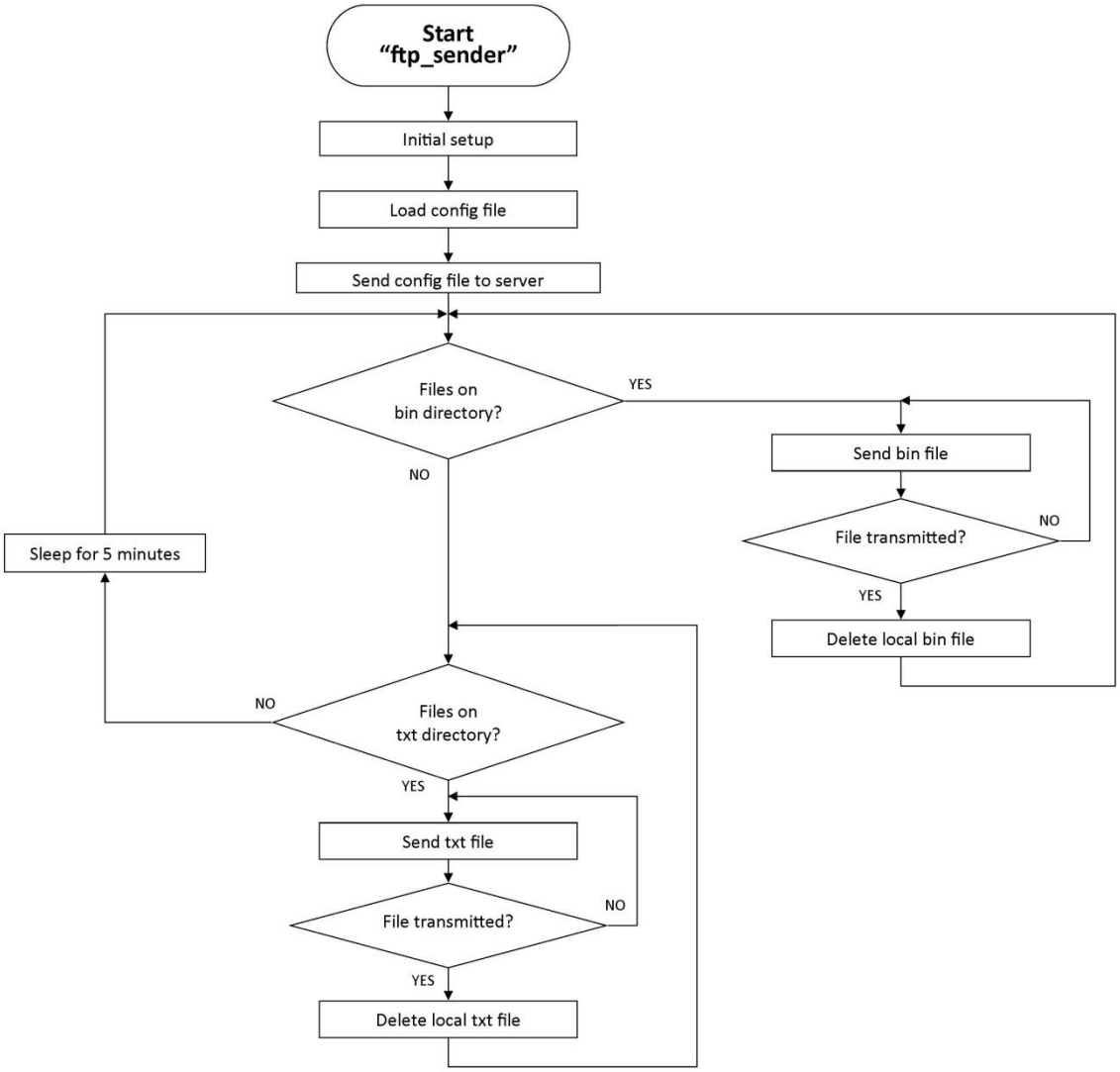


Figura 3.17 - Diagramma di flusso dell'eseguibile "ftp_sender".

Dopo una fase di setup iniziale, viene caricato il contenuto del “config file”, per permettere all’applicativo di conoscere il nome della macchina su cui sta girando, e quindi, potersi identificare con il server quando gli trasferisce i file. Poi questo file di configurazione viene spedito al server, in modo che si possano avere a disposizione gli offset del singolo sito, nel caso siano richieste elaborazioni a posteriori dei dati acquisiti.

A questo punto, viene verificata la presenza di file binari o txt nelle rispettive cartelle locali e nel caso fossero presenti vengono trasferiti, uno alla volta, in delle specifiche directory sul server tramite una connessione FTP. Come si vedrà nel paragrafo seguente, sul server, ogni sito ha una propria directory principale in cui deve trasferire i propri file. Dopo essere stato trasmesso, il file, viene cancellato dalla memoria locale.

Terminati tutti i file da trasmettere, l'applicativo va in uno stato di "sleep" per 5 minuti, riducendo quasi del tutto l'utilizzo di risorse hardware, per poi rientrare nel loop e ri-controllare la presenza di file che nel frattempo il "capture" dovrebbe aver generato.

Durante l'esecuzione di questo applicativo vengono adottate alcune precauzioni:

- viene verificata l'effettiva riuscita del trasferimento sul singolo file (sia sul config file, che sui bin, che sui txt) e, nel caso non sia andata a buon fine, sono previsti 3 tentativi di ritrasmissione prima di passare al file successivo. Inoltre, nel caso non si sia riusciti nel trasferimento, il file non viene cancellato. (questo potrebbe accadere in caso di problemi di connessione col server);
- viene verificata l'effettiva cancellazione del file dopo la sua trasmissione e nel caso non sia andata a buon fine viene segnalato in un file di report (questo potrebbe accadere nel caso ci fossero altri servizi che stanno accedendo al file nello stesso momento, come ad esempio il "capture" mentre sta generando un bin o un txt);
- la connessione col server viene instaurata solo quando necessaria e non viene mantenuta per tutto il periodo di esecuzione del programma (che si ricorda è in esecuzione continua), in modo da evitare "time-out" da parte del server;
- i file che devono essere trasferiti vengono aperti in modalità di "sola lettura", quindi, nel caso si dovessero verificare anomalie o crash durante l'esecuzione di "ftp_sender", i file non verrebbero modificati ne corrotti.

Dividendo i compiti in questi 3 applicativi si hanno diversi vantaggi, tra i quali:

- Separando la parte di configurazione della macchina dal programma "capture" si alleggerisce quest'ultimo, che altrimenti dovrebbe caricare in memoria anche la parte per la configurazione benché non venga usata;
- La configurazione del dispositivo, che è un'operazione particolare, non può essere lanciata per errore perché va lanciata appositamente;
- Le applicazioni principali ("capture" e "ftp_sender") sono abilitate ad aprire il file di configurazione in modalità di sola lettura e quindi non sono in grado di modificarlo, neanche per errore o anomalia;

- Separando la spedizione dall'elaborazione, in caso di problemi col server o con la connessione dati, che potrebbero rallentare enormemente il processo di trasferimento dei file, si elimina la possibilità che il sistema non sia pronto a ricevere i nuovi dati dalla scheda di acquisizione. Il "capture" essendo esonerato dal gestire l'invio dei file ed avendo priorità di esecuzione maggiore, sarà sempre pronto a ricevere i nuovi dati;
- In caso di aggiornamento di una parte del software si può sostituire solo l'applicativo influenzato, lasciando gli altri in esecuzione.

Inoltre, durante l'esecuzione di questi programmi, se dovessero riscontrarsi problemi o anomalie, è prevista la generazione di un file dettagliato di report per segnalare appunto le anomalie riscontrate. Ad esempio, nel report di "ftp_sender" verrebbero segnalati i file che l'applicativo non è riuscito a trasferire e quali sono state le problematiche sulla connessione, mentre in quello di "capture" verrebbero segnalate le anomalie sulla comunicazione con la scheda. Questi file di testo sono di grande utilità, in quanto permettono al manutentore di capire la causa del problema e quindi di poterla risolvere.

3.2.2 Lato server

Per come è stata progettata l'infrastruttura di monitoraggio, il server è un componente che riveste un ruolo fondamentale. Questo per tre motivazioni principali:

- Riceve i dati da tutti i siti di monitoraggio, concentrandoli tutti in un'unica sede (il che semplifica le operazioni di classificazione, memorizzazione, backup, visualizzazione ed elaborazioni a posteriori) e libera memoria sui uPC, che tra l'altro sono quelli più a rischio di crash visto gli ambienti elettromagneticamente ostili in cui vengono installati;
- Implementa un sito web per visualizzare da browser tutti i parametri di Power Quality acquisiti dai diversi apparati;
- Scambia continuamente "informazioni di servizio" con i vari siti, permettendo di tenere sotto controllo il loro stato e di poterli manutenere da remoto se necessario (ciò sarà più chiaro nel successivo paragrafo quando verrà trattato il sistema di gestione e controllo).

Per quanto riguarda il primo e secondo punto, in Figura 3.18 viene mostrata una schematizzazione funzionale del server.

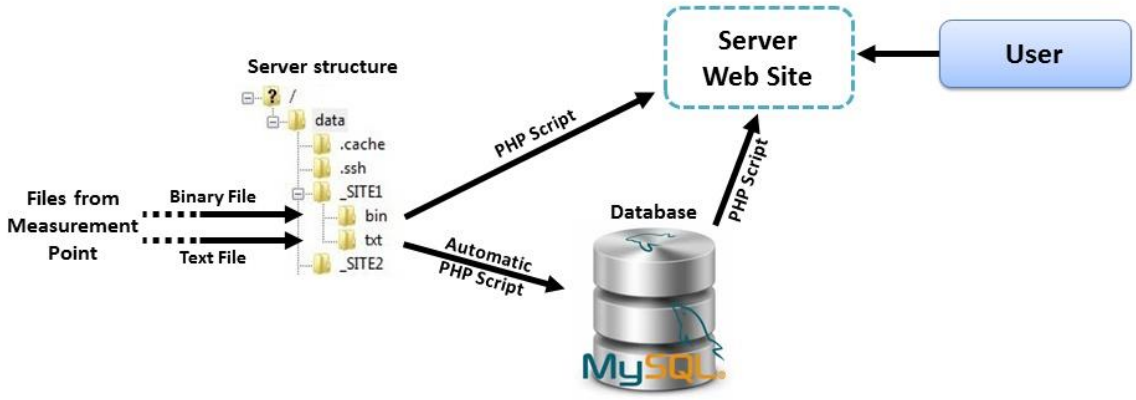


Figura 3.18 - Struttura del server.

Il Server è strutturato in modo da avere una directory dedicata per ogni punto di monitoraggio. Quindi i siti, tramite la connessione FTP, caricano continuamente i file binari e di testo nelle rispettive cartelle. Sul server ad intervalli regolari, attualmente ogni 5 minuti, vengono eseguiti automaticamente degli script PHP che leggono il contenuto della cartella txt e popolano il database con il contenuto di questi txt, cancellando poi i file di testo. A questo punto, tutte le informazioni sono memorizzate nel DB. Quando un utente di internet naviga sulle pagine presenti sul sito web, questo, tramite altri script PHP, si collega sia al DB sia alla cartella contenente i file binari e li visualizza sullo schermo dell'utente, dando anche la possibilità di scaricarli sul proprio computer.

Nei seguenti sotto-paragrafi, verrà descritta dapprima la piattaforma software implementata nel server e poi verranno mostrati alcuni script ed interfacce web per la memorizzazione e visualizzazione dei dati di monitoraggio.

3.2.2.1 Sistema Operativo e servizi importanti

Dopo aver acquistato l'hardware (paragrafo 3.1.6), lo si è formattato, per eliminare ogni preesistente software e poi si è passati all'installazione del sistema operativo e di tutti i moduli necessari.

Come S.O. si è scelta la distribuzione Linux senza interfaccia grafica (no GUI – solo CLI) di Ubuntu server, per le sue caratteristiche principali:

- Free e open source;
- Sicuro, non presenta porte aperte dopo l'installazione;
- Stabile e aggiornato, utilizzata la versione LTS (Long Term Support);
- Semplice da utilizzare e configurare;
- Simile, per certi aspetti, alla distribuzione sui siti di monitoraggio;
- Con una grande community di utilizzatori ed appassionati, che agevola il ritrovamento di informazioni per il supporto tecnico in caso di bisogno.

Alla fine di una lunga serie di passaggi, per scegliere, installare e configurare il S.O. ed i vari servizi utili per l'infrastruttura di monitoraggio, si è arrivati ad avere un server LAMP (Linux-Apache-MySQL-PHP/Perl) con i seguenti moduli:

- Apache, per il Web server;
- MySQL, come gestore di database;
- PHP, come linguaggio di scripting;
- phpMyAdmin, per amministrare graficamente e manualmente i DB creati in MySQL;
- FTP server, per gestire le varie connessioni in ingresso dai vari punti di monitoraggio;
- Demone NTP configurato con i server dell'I.N.Ri.M., per avere l'orologio di sistema sempre sincronizzato con questi ultimi (molto utile soprattutto nell'eseguire script automaticamente con attività pianificate);
- Firewall, per cercare di proteggere la struttura da attacchi di malintenzionati;
- Webmin ed SSH client/server per la gestione e la manutenzione da remoto della macchina e dei vari siti.

Inoltre, per poter essere accessibile dai vari sistemi di monitoraggio, indipendentemente dalla loro posizione, lo si è configurato con un indirizzo IP fisso e pubblico sulla rete internet.

3.2.2.2 Applicativi e script sviluppati per il monitoraggio

Sulla piattaforma software realizzata, che si è appena descritta, sono inoltre stati sviluppati vari script PHP e diverse pagine HTML/JavaScript, fondamentali per il funzionamento dell'intera infrastruttura e per la visualizzazione dei dati di Power

Quality. Alcuni script vengono lanciati automaticamente ad intervalli di tempo regolari, mentre altri, vengono eseguiti solo durante la navigazione di un utente sul sito web, quando necessari.

In Figura 3.19 è mostrato il principio di funzionamento di uno degli script PHP principali.

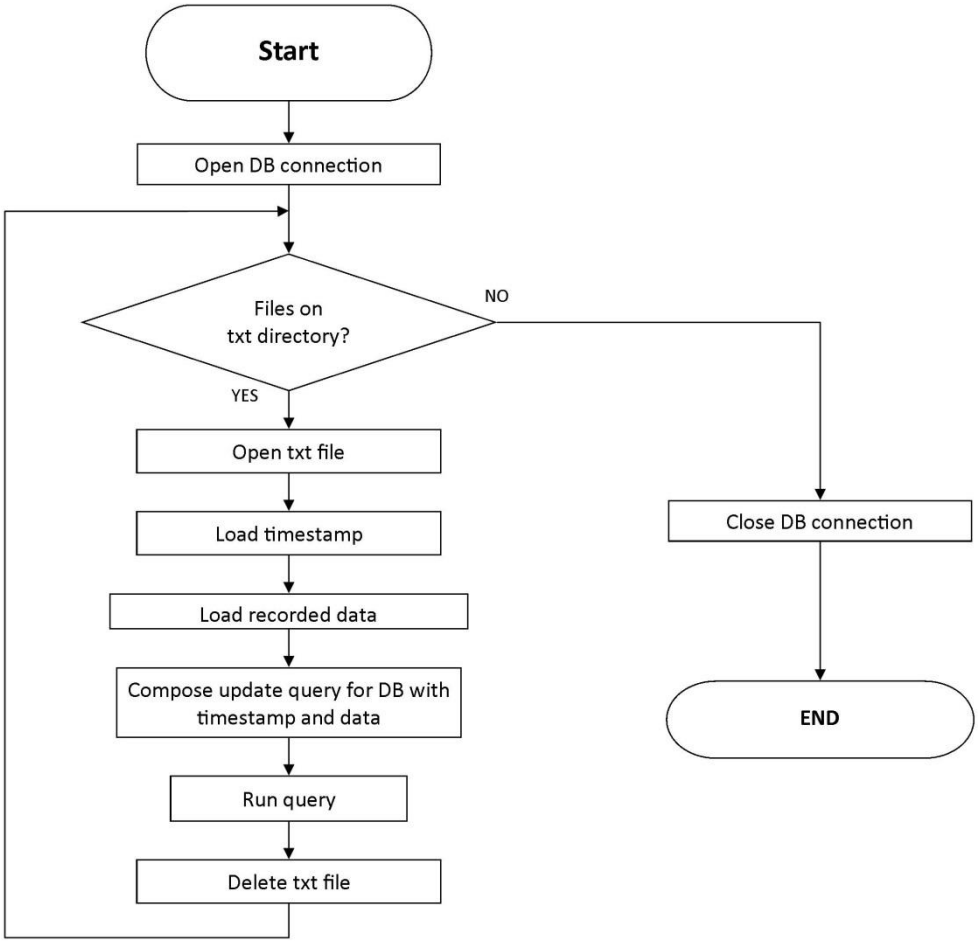


Figura 3.19 - Diagramma dello script principale per il popolamento del DB.

Questo script viene lanciato automaticamente a cadenze prefissate (ogni 5 minuti) e per ogni cartella txt (una per ogni sito di monitoraggio). Come si può vedere anche dal diagramma molto esplicativo, questo processo si occupa di popolare il database con i dati contenuti in tutti i file txt ricevuti fino al momento della sua esecuzione. Infine, se i dati sono stati inseriti correttamente nel DB, i txt vengono cancellati. (Il codice relativo a questo script si può trovare in Appendice).

Come si è già detto, sul server, oltre ai vari script, sono state realizzate diverse interfacce web per la visualizzazione dei dati di PQ. Verranno quindi ora mostrate le schermate principali (tralasciando il codice per generarle) che un utente può trovare quando si collega al sito web implementato sul server.

Una delle interfacce web principali del sito è quella mostrata in Figura 3.20. Selezionando il sito di interesse (nell'esempio Roma-Corviale) e il periodo di osservazione, si può vedere l'andamento delle grandezze monitorate, in questo caso la frequenza di rete. Questa interfaccia, basata fortemente su JavaScript, va ad interpellare il database utilizzando degli script PHP nascosti e ne visualizza i dati ricevuti. Passando con il mouse sopra il grafico, viene visualizzando anche il valore puntuale ed il timestamp di ogni acquisizione in corrispondenza della posizione del puntatore. E' prevista anche la possibilità di interagire con il grafico, selezionando ed ingrandendo alcune sezioni.

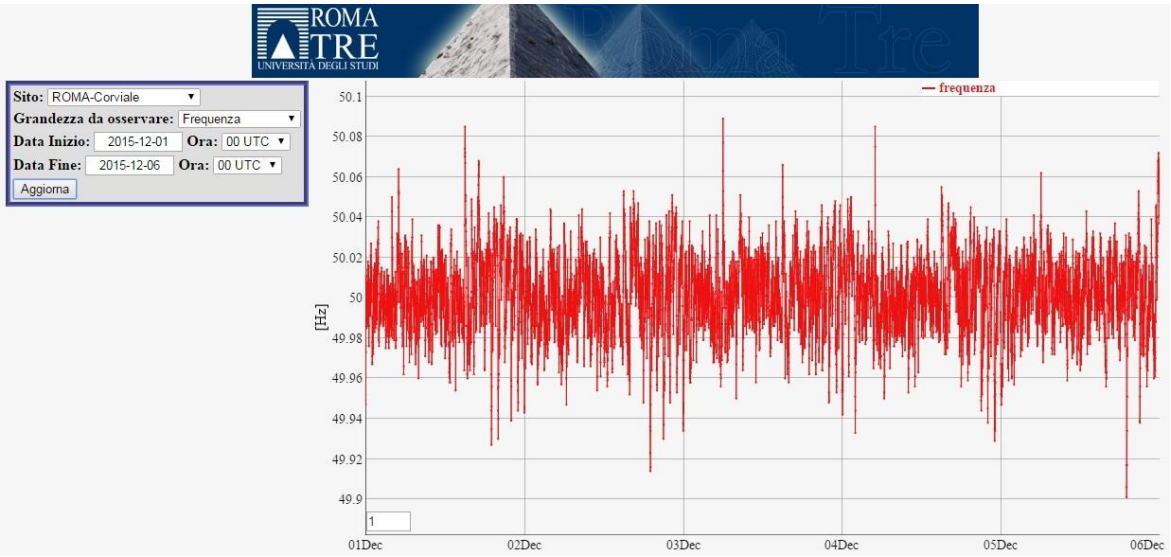


Figura 3.20 - Schermata principale del sito per l'andamento della frequenza.

Nella stessa pagina HTML è possibile selezionare dal menu la voce "tensione efficace" e verrà visualizzato l'andamento dell'ampiezza efficace delle tre fasi R-S-T del sistema trifase sotto osservazione. Come mostrato in Figura 3.21 per il sito di Corviale nel periodo compreso tra il 1/12 ed il 6/12 del 2015 (orario 00:00 UTC).

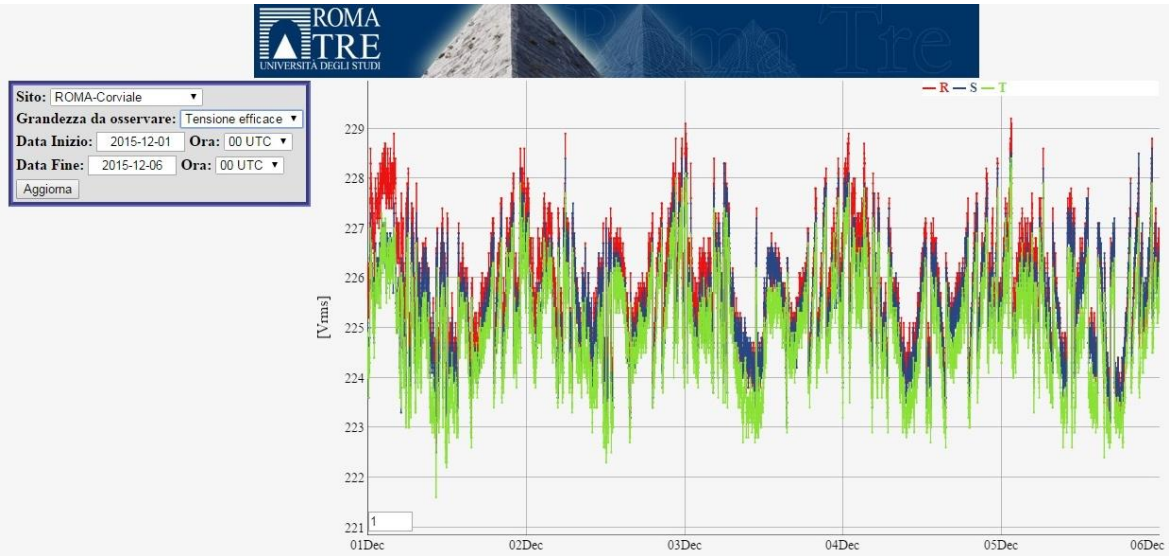


Figura 3.21 - Schermata principale del sito per l'andamento delle tensioni efficaci delle tre fasi.

Sempre agendo sullo stesso menu a tendina, è possibile scegliere di visualizzare l'andamento dell'ampiezza delle varie armoniche stimate per quel periodo in quel sito. Ad esempio, in Figura 3.22 è visualizzato l'andamento dell'ampiezza dell'undicesima armonica calcolata su ogni fase (sempre per il periodo ed il sito delle schermate precedenti).

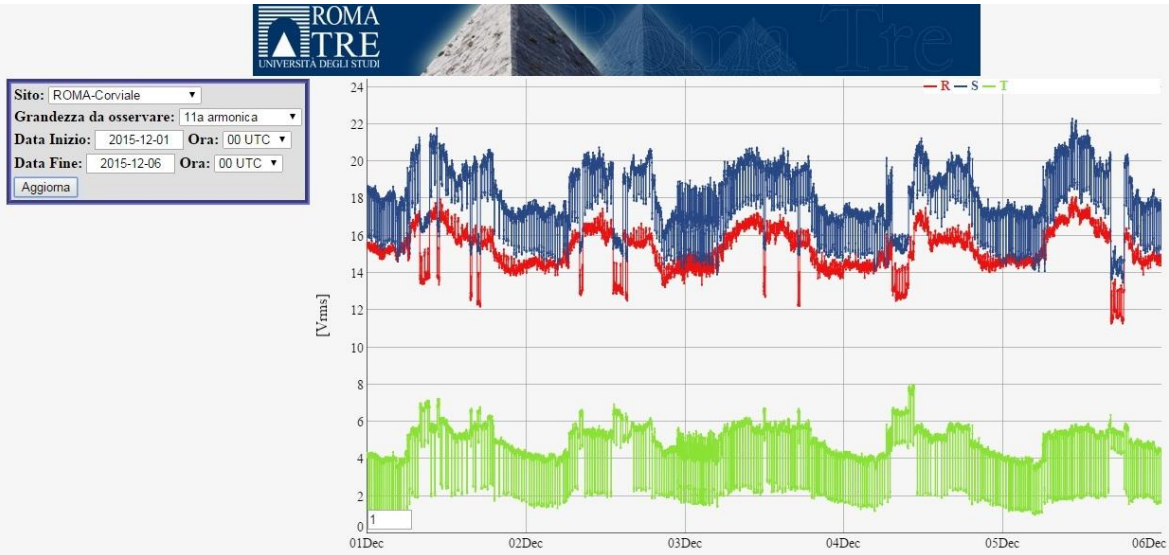


Figura 3.22 - Schermata principale del sito per l'andamento dell'ampiezza delle armoniche.

Inoltre, da una delle schermate precedenti, cliccando con il tasto destro del mouse su un punto del grafico, si apre una nuova finestra (vedi Figura 3.23) in cui viene visualizzato:

- il file binario con i valori “grezzi” provenienti direttamente dall’ADC della scheda di acquisizione;
- la composizione armonica del segnale acquisito;
- lo sfasamento calcolato tra le fasi R-S-T;
- un link per scaricare il file bin sul computer dell’utente per eventuali elaborazioni ed analisi.

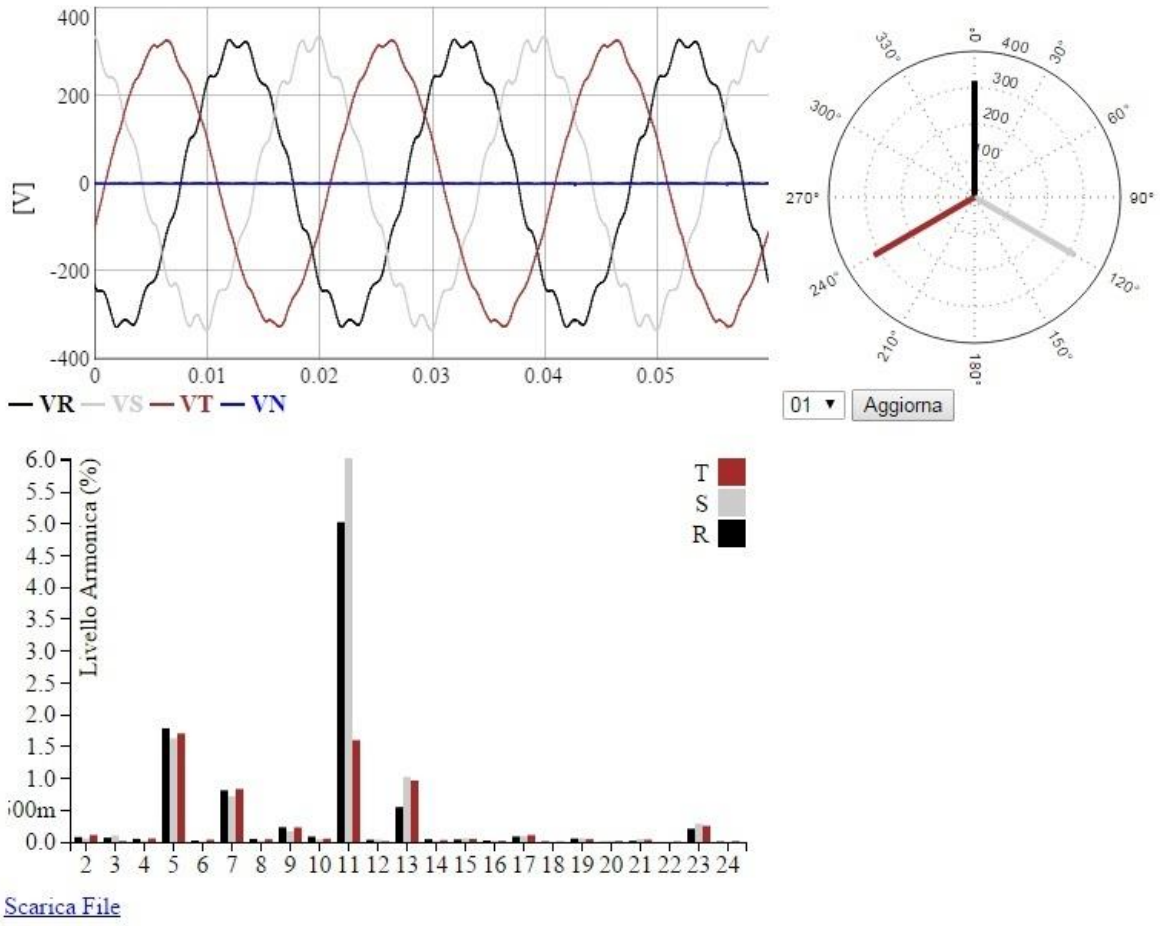


Figura 3.23 - Schermata del sito per visualizzare i dati grezzi acquisiti dall'ADC e la composizione armonica.

3.3 Il Sistema di Gestione e Controllo

In questo paragrafo verranno mostrati l'intelaiatura software ed i componenti che sono stati sviluppati per permettere agevolmente di gestire e controllare tutte le macchine dell'infrastruttura da remoto.

3.3.1 SSH e Webmin

Essendo tutta l'infrastruttura di monitoraggio costituita da macchine Linux, si è deciso di utilizzare, per il controllo e la gestione dei vari sistemi remoti, il protocollo SSH e l'applicativo Webmin.

L'SSH (Secure SHell) è un protocollo di rete, con interfaccia a riga di comando, che permette di stabilire una sessione remota sicura con un altro pc di una rete informatica. "Sicura" perché l'intera comunicazione, sia l'autenticazione (mutua) che la sessione di lavoro, avviene in maniera cifrata. Inoltre, ha la caratteristica di essere gratuito.

Con queste caratteristiche, questo protocollo è ormai diventato uno standard di fatto per l'amministrazione remota di sistemi Linux e UNIX, ed è per questo motivo che è stato scelto ed è stato installato sui vari device (di base sui BeagleBone Black si avrebbe avuto solo l'SSH client per connessioni in uscita, quindi si è dovuto installare anche l'SSH server per permettere connessioni in entrata).

In Figura 3.24 è mostrato uno screenshot di una sessione SSH con un uPC di uno dei probe, durante un'operazione di manutenzione.

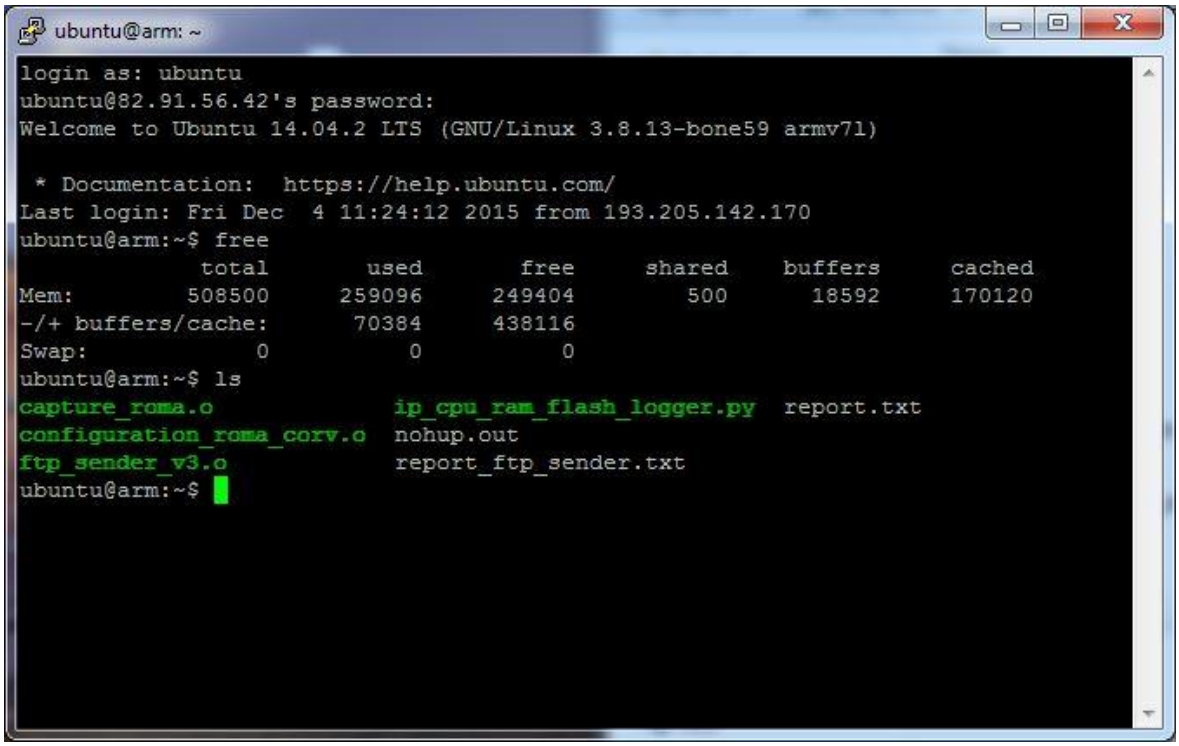


Figura 3.24 - Screenshot di una connessione SSH con uno dei siti (Roma - Corviale).

Come si può notare, con questo tipo di connessione si ha il pieno controllo del sistema. (In figura, inoltre, si possono vedere: gli applicativi descritti al paragrafo precedente, i file di report generati ed un altro script che verrà mostrato più avanti.)

Webmin, invece, è un software (sempre gratuito) che una volta installato su una macchina Linux o Unix permette di amministrarla da remoto tramite un'interfaccia web, quindi da browser. Permette di gestire svariati aspetti del sistema, sia da un punto di vista hardware che software, inoltre, utilizza una connessione cifrata HTTPS (HyperText Transfer Protocol over Secure Socket Layer), usata comunemente per garantire trasferimenti riservati di dati nel web, come per i siti bancari e di pagamento on-line.

Praticamente permette di eseguire le stesse operazioni che si potrebbero fare durante una sessione SSH, ma graficamente e da browser. In Figura 3.25 è mostrata una schermata di una connessione ad un sito con Webmin.

Essendo questo software con interfaccia grafica, ovviamente, la sua esecuzione appesantisce un po' il BeagleBone, di conseguenza, solitamente viene lasciato disattivato sui uPC e si cerca di utilizzare solo l'SSH per la manutenzione, ma in caso di necessità viene avviato in modo da potercisi connettere anche da browser.

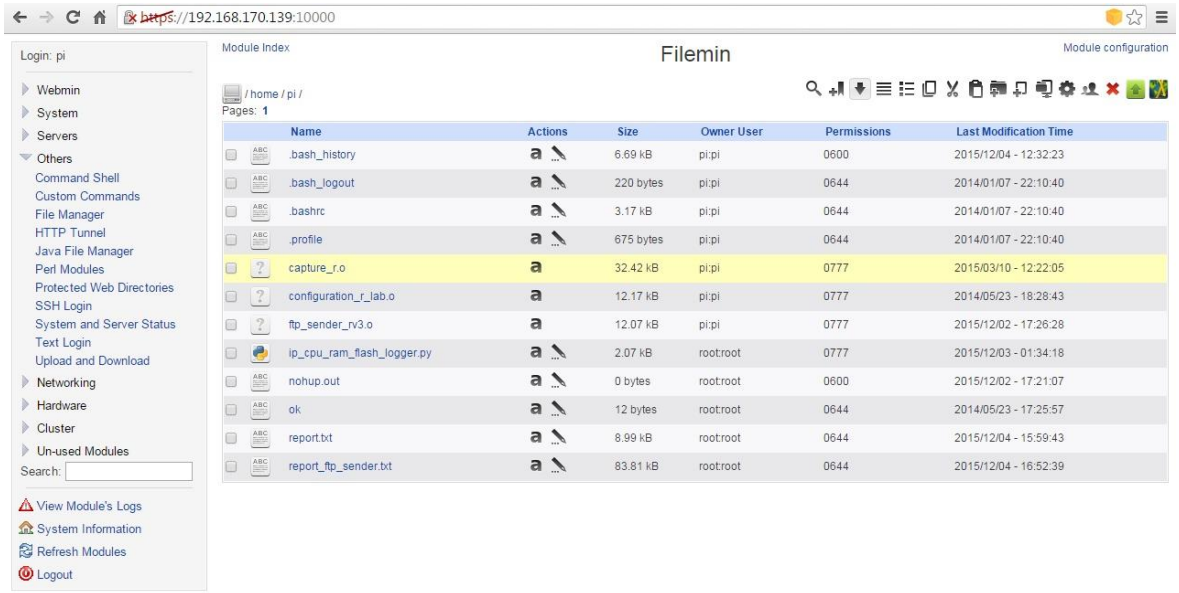


Figura 3.25 - Schermata di Webmin di un sito (Roma - Lab).

3.3.2 "Port forwarding" e "reverse-tunneling"

Quindi, ricapitolando, sia l'SSH che Webmin sono strumenti appositamente studiati per gestire macchine Linux da remoto, sono gratuiti e sicuri perché utilizzano protocolli di comunicazione cifrati. Inoltre, non vincolano ad usare sistemi Linux per connettersi ai dispositivi, ma possono essere utilizzati anche da macchine Windows, Mac OS, Unix e da Smartphone.

Per potersi collegare ai siti di monitoraggio, però, questi strumenti **necessitano dell'indirizzo IP della macchina da controllare**. Questo, per il server non è un problema perché, come si è già detto, è stato configurato con un IP fisso e pubblico, ma per i vari uPC dislocati sul territorio la situazione è differente. I BeagleBone, infatti, vengono nascosti dall'IP del modem, che in aggiunta è anche un IP "dinamico" e quindi può cambiare continuamente e senza preavviso.

Per ovviare a questo "inconveniente" degli IP dei probe, sono stati studiati e testati alcuni metodi:

- comunicazione degli Ip dei siti al server;
- apertura porte sul modem/router;
- reverse-tunneling tramite il server.

Nell'infrastruttura operante nel comune di Roma, avendo a disposizione una connessione ADSL dedicata per ogni sito ed avendo a disposizione dei modem/router con buone doti di configurabilità, si è scelto di utilizzare la tecnica che prevede l'uso dei primi due punti.

Come è già stato mostrato al paragrafo 3.1.4 inerente il modem, avendone la possibilità, i router sono stati tutti configurati con il port forwarding sulle porte 22 (per l'SSH) e 10000 (per Webmin). Ciò permette, ad un utente di internet che cerca di collegarsi a una di queste porte del modem di un sito, di essere rediretto al BeagleBone sulle stesse porte, dando così la possibilità all'utente di stabilire una sessione remota con il uPC.

Per conoscere l'indirizzo IP dei modem dei vari siti, a cadenze temporali prestabilite (attualmente ogni 10 minuti), sui vari BeagleBone Black vengono lanciati in automatico due script che richiamano una pagina PHP del web server e gli forniscono alcuni parametri insieme al nome della macchina che sta effettuando la chiamata. La pagina PHP è collegata ad un database e quando viene chiamata registra in una tabella:

- il nome della macchina chiamante, per identificare il sito;
- l'indirizzo IP di provenienza, che è quello del modem a cui bisogna puntare per collegarsi con i uPC;
- data e ora della chiamata, per vedere quanto sono aggiornati i dati;
- gli altri parametri che gli vengono passati.

In Figura 3.26 è mostrato uno schema di funzionamento del metodo appena descritto.

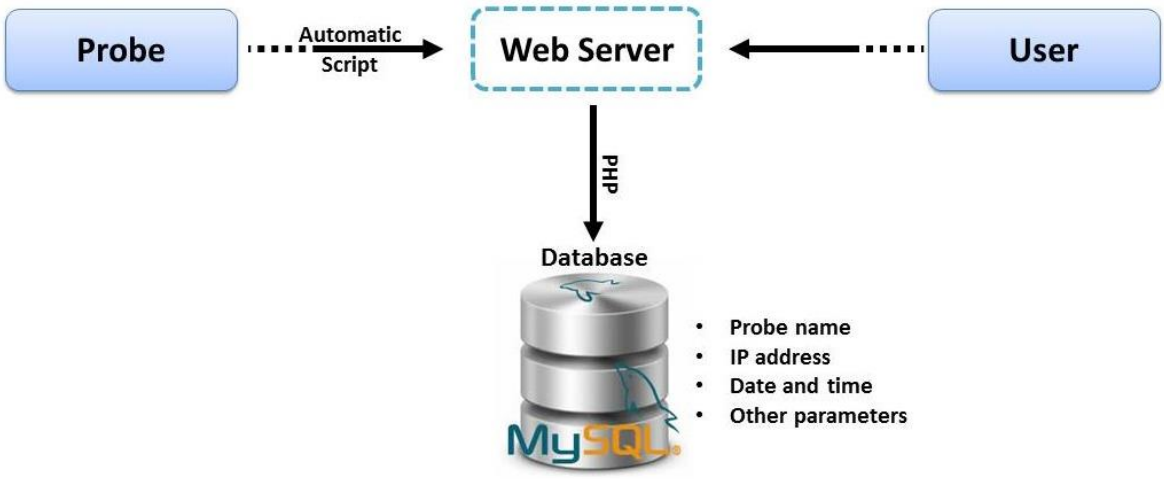


Figura 3.26 - Schema di principio per la registrazione degli IP dei siti.

A questo punto, avendo un database facilmente accessibile dove poter recuperare l'indirizzo IP di ogni sito, l'utente non ha più problemi a collegarsi ad uno dei probe per fare manutenzione e può collegarcisi in maniera diretta, utilizzando un terminale SSH o mediante l'interfaccia di Webmin (Figura 3.27).

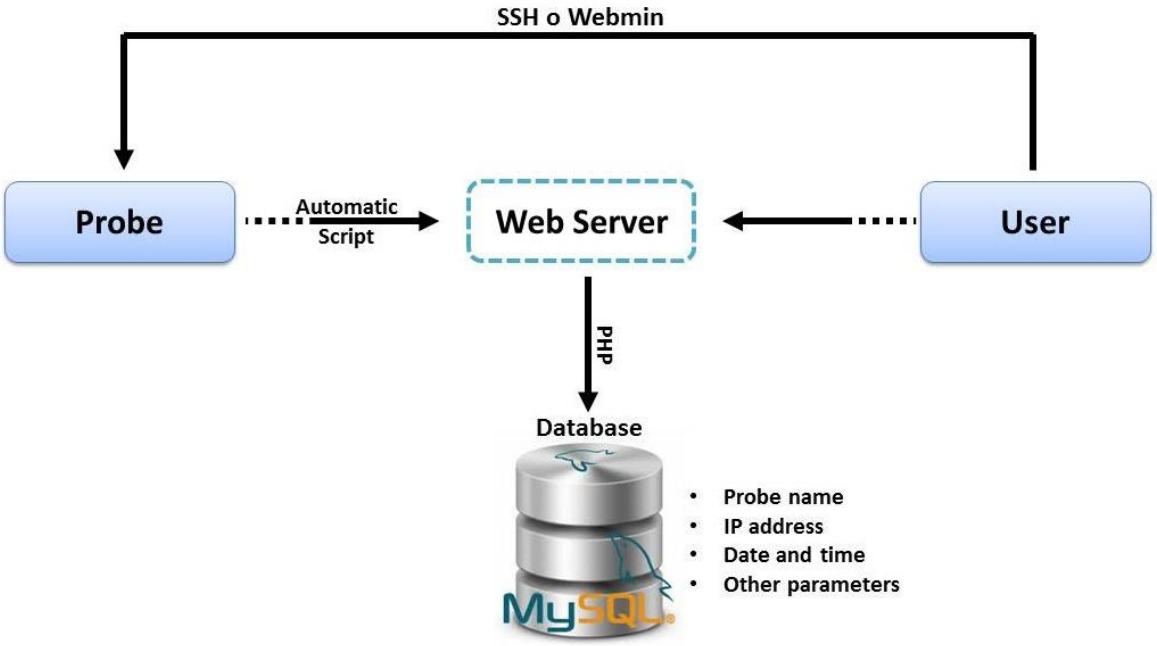


Figura 3.27 - Schema di principio del collegamento ai probe distribuiti sul comune di Roma.

Nel caso invece, non sia possibile impostare il port forwarding sulle porte del modem/router, come ad esempio quando si utilizza una connessione ADSL con un modem Wi-Fi o Ethernet non configurabile, o si utilizza un modem 3G/4G, si può fare "reverse-tunneling" tramite il server per collegarsi ai dispositivi remoti.

Il "reverse-tunneling" prevede che il BeagleBone mantenga continuamente una connessione con una porta sul server, l'utente che si collega al server su quella porta verrà reindirizzato ad una specifica porta sul BeagleBone e tutti i comandi transiteranno attraverso il server (Figura 3.28). Questa tecnica fa uso del protocollo SSH, quindi le connessioni che vengono instaurate sono cifrate ed inoltre permette di far transitare su questi collegamenti anche applicazioni diverse (come ad esempio Webmin) incapsulandole nel tunnel SSH.

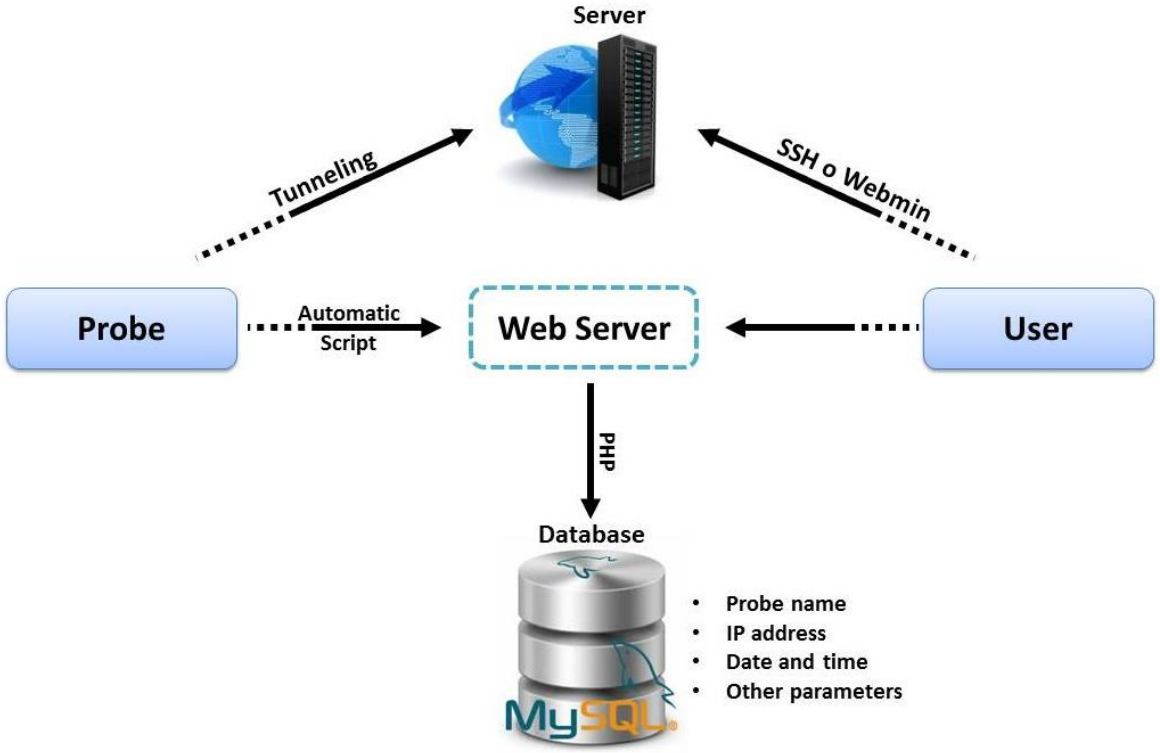


Figura 3.28 - Schema di principio per un collegamento ai probe tramite reverse-tunneling.

Per un collegamento di questo tipo, è necessario sul BeagleBone Black uno script che si occupi di mantenere sempre instaurata una connessione col server e di ripristinarla nel caso si interrompesse. Ciò perché, in assenza di questo collegamento, il uPC sarebbe irraggiungibile. La connessione che deve essere mantenuta viene instaurata utilizzando il comando:

ssh -N -R 2222:localhost:22 serverUser@11.11.11.11

dove "2222" è la porta del server, "22" è la porta del uPC e "11.11.11.11" è l'indirizzo IP pubblico del server.

Nell'infrastruttura di Roma si è scelto di usare il metodo con le porte aperte perché, rispetto al reverse-tunneling:

- permette un controllo più diretto dei sistemi;
- le connessioni vengono instaurate solo quando necessario, e cioè molto di rado;
- non c'è il rischio di "perdere" il uPC se la connessione cade;
- i modem in dotazione lo permettevano.

Anche nel caso si utilizzasse il reverse-tunneling, la conoscenza degli indirizzi IP dei vari siti è molto importante per riconoscere su quale porta del server è collegato un determinato sistema.

Tornando alla Figura 3.26, per poter registrare gli IP nel database, nei BeagleBone vengono eseguite due operazioni distinte:

- viene lanciato, ogni 30 minuti, un semplice **comando da shell** inserito nel crontab:

  curl --silent http://ip.server/ip_logger.php?sito=site_name

  questo, facendo uso del servizio "curl" installato sui uPC, richiama la pagina PHP "ip_logger", sul server all'indirizzo "ip.server", e gli passa solo il parametro identificativo del sito "site_name";
- viene lanciato uno **script Python**, ogni 10 minuti, che richiama sempre la stessa pagina sul server ma passandogli, oltre al nome, anche altri parametri di funzionamento del uPC. Anch'esso è stato inserito nel crontab, come:

  python /home/ubuntu/ip_cpu_ram_flash_logger.py site_name

Sebbene sia sufficiente lo script Python per far recepire l'IP al server, è stato inserito anche il primo comando di "curl", in modo che se per qualche ragione (un aggiornamento andato a male, una corruzione dello script, ecc) l'interprete Python non riuscisse più ad eseguire lo script, il server riceverebbe comunque l'IP del probe. Questa "ridondanza" nel fornire l'IP al server è dovuta all'estrema importanza di questo indirizzo per potersi collegare ai sistemi e correggere malfunzionamenti ed anomalie.

Lo script Python realizzato, il cui diagramma di funzionamento è mostrato in Figura 3.29 ed il cui codice si può trovare in Appendice, ha anche il compito di fornire alcuni parametri di funzionamento del BeagleBone al server (quali utilizzo medio della CPU, della RAM e della Flash), con lo scopo di prevenire eventuali anomalie, come si vedrà dall'interfaccia mostrata successivamente nel testo (paragrafo 3.3.4).

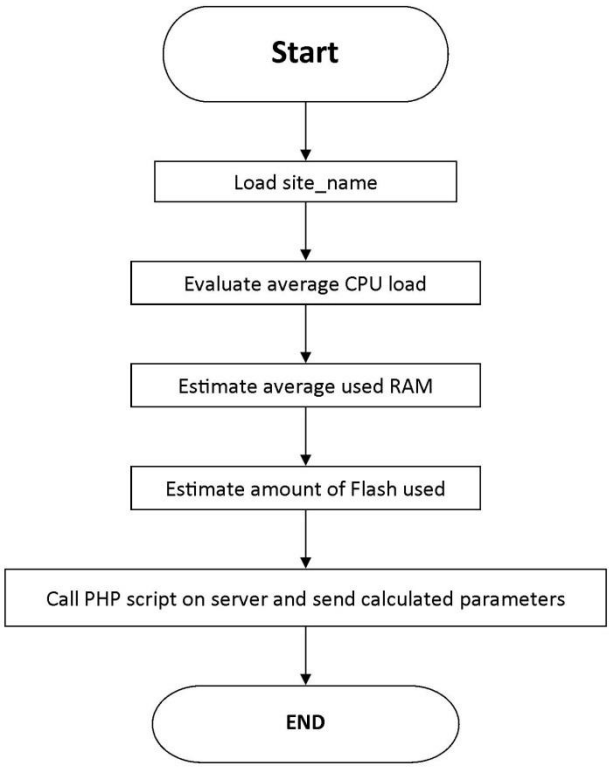
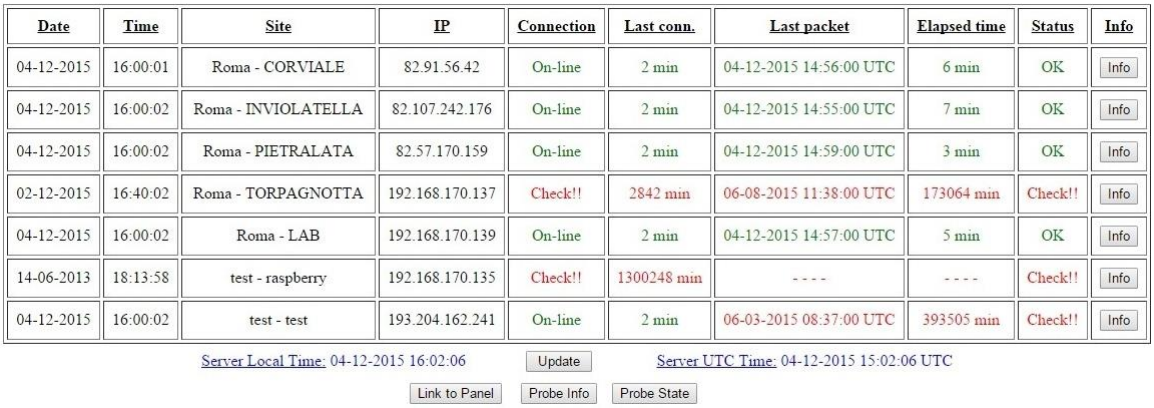


Figura 3.29 - Diagramma di flusso dello script Python implementato sul BeagleBone Black.

Verranno ora mostrate e discusse le interfacce del sistema di gestione e controllo dell'infrastruttura residenti sul server, per il codice relativo si rimanda in Appendice. Queste pagine web sono state create appositamente per agevolare ed in alcuni casi permettere, le operazioni di manutenzione sui siti.

3.3.3 L'interfaccia "ip_table"

Questa è una delle schermate di controllo principali e permette di tenere sott'occhio lo stato dell'intera infrastruttura (Figura 3.30). E' stata realizzata con codice misto PHP e Javascript e va a recuperare le informazioni dai vari database nel server (sia da quello di manutenzione che da quello di monitoraggio).



Date	Time	Site	IP	Connection	Last conn.	Last packet	Elapsed time	Status	Info
04-12-2015	16:00:01	Roma - CORVIALE	82.91.56.42	On-line	2 min	04-12-2015 14:56:00 UTC	6 min	OK	Info
04-12-2015	16:00:02	Roma - INVIOLATELLA	82.107.242.176	On-line	2 min	04-12-2015 14:55:00 UTC	7 min	OK	Info
04-12-2015	16:00:02	Roma - PIETRALATA	82.57.170.159	On-line	2 min	04-12-2015 14:59:00 UTC	3 min	OK	Info
02-12-2015	16:40:02	Roma - TORPAGNOTTA	192.168.170.137	Check!!	2842 min	06-08-2015 11:38:00 UTC	173064 min	Check!!	Info
04-12-2015	16:00:02	Roma - LAB	192.168.170.139	On-line	2 min	04-12-2015 14:57:00 UTC	5 min	OK	Info
14-06-2013	18:13:58	test - raspberry	192.168.170.135	Check!!	1300248 min	- - - -	- - - -	Check!!	Info
04-12-2015	16:00:02	test - test	193.204.162.241	On-line	2 min	06-03-2015 08:37:00 UTC	393505 min	Check!!	Info

Server Local Time: 04-12-2015 16:02:06 Update Server UTC Time: 04-12-2015 15:02:06 UTC

Link to Panel Probe Info Probe State

Figura 3.30 - Interfaccia web "ip_table".

Nella tabella viene mostrato:

- il nome associato al sito di monitoraggio;
- l'indirizzo IP per collegarsi al probe in caso di necessità;
- la data e l'ora dell'ultimo aggiornamento dell'IP da parte del sistema remoto (a sinistra);
- lo stato della connessione con il uPC;
- il tempo trascorso dall'ultima connessione avvenuta con il probe. Riguarda gli script che vengono lanciati automaticamente, ogni 10 minuti, sui BeagleBone, quindi se passano più di 11 minuti ci deve essere qualche problema di connessione;
- l'ultimo pacchetto di dati per il monitoraggio ricevuto;
- il tempo trascorso dalla ricezione di quest'ultimo. E' previsto che il programma "ftp_sender" sui probe spedisca i file ad intervalli regolari di 5 minuti, quindi, superato questo tempo dall'ultimo pacchetto, significherebbe che potrebbero esserci problemi di connessione, problemi sull'esecuzione di qualche applicativo sul uPC o problemi sulla scheda di acquisizione;
- un indicatore generale dello stato del sito;
- un pulsante che ridirige all'interfaccia "code_panel" specifica per il singolo sito.

Se vengono riscontrate anomalie, sul tempo trascorso dall'ultima connessione o dall'ultimo file ricevuto, la riga da verde diventa rossa e viene segnalato all'operatore di controllare il probe (Check!!).

Inoltre, in basso, viene visualizzato il timestamp corrente dell'orologio sul server in UTC (a destra) e in formato locale (a sinistra). Si ricorda che questo orario è sincronizzato con i server dell'I.N.Ri.M. tramite un servizio NTP che gira continuamente sulla macchina.

Infine, vi sono 4 pulsanti con le seguenti funzioni:

- "Update", per ricaricare la pagina con gli ultimi valori memorizzati nei DB;
- "Link to Panel", per essere rediretti alla pagina principale del sito per la visualizzazione dei dati di monitoraggio (Figura 3.20);
- "Probe Info", per caricare l'interfaccia "code_panel" che verrà mostrata più avanti;
- "Probe State", per aprire la pagina web "probe_state", mostrata anch'essa nel seguito.

Guardando la figura, si può vedere che il sito "Roma – Torpagnotta" è rosso, infatti per un certo periodo di tempo ci sono stati problemi con la connessione ADSL in quel punto ed il uPC, quindi, è stato portato in laboratorio dove gli è stato assegnato l'indirizzo "*192.168.170.137*" mostrato nella tabella.

3.3.4 L'interfaccia "probe_state"

Questo pannello di controllo (mostrato in Figura 3.31), invece, serve per avere sottocchio la situazione dei vari parametri di funzionamento di tutti i probe, acquisiti utilizzando lo script Python analizzato nel paragrafo precedente e visto in Figura 3.29.

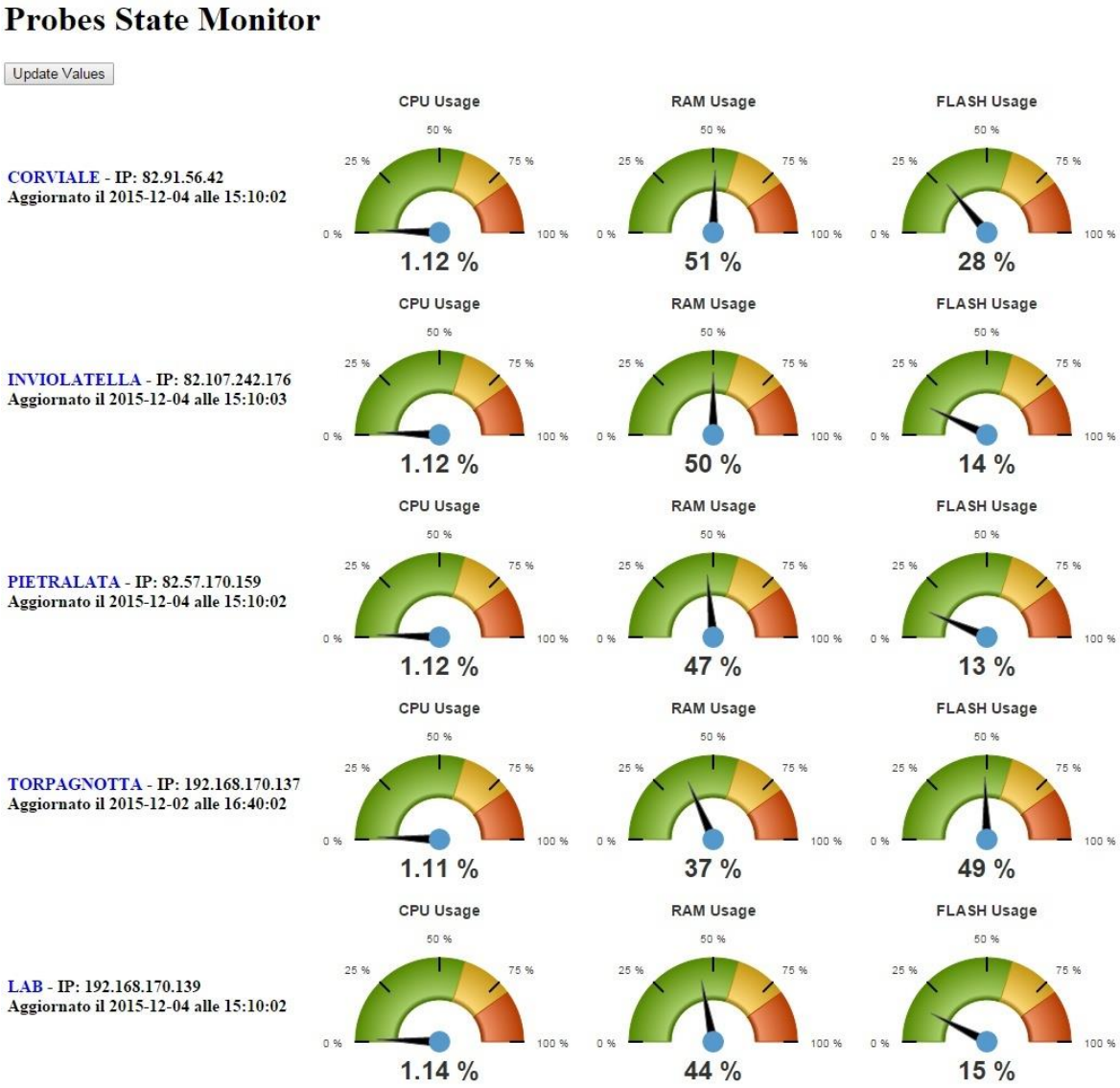


Figura 3.31 – Schermata della pagina web "probe_state".

Seguendo i parametri monitorati, ovvero:

- utilizzo della CPU, mediato negli ultimi 15 minuti;
- uso della memoria RAM;
- occupazione della memoria Flash;

è possibile capire se si sta verificando qualche anomalia a livello software sul BeagleBone Black e quindi intervenire prima che sia troppo tardi.

Anche in questa interfaccia viene mostrato l'indirizzo IP dei vari sistemi e l'ultima connessione avvenuta, in modo da verificare quanto i dati visualizzati siano aggiornati e se il sistema li sta ancora trasmettendo o sta avendo qualche problema.

Durante lo screenshot, tutti i sistemi, tranne ovviamente "Torpagnotta" che si trovava nella stessa situazione descritta al paragrafo precedente, avevano in esecuzione i vari

applicativi per il monitoraggio e l'interfaccia di Webmin, il che giustifica un utilizzo della RAM intorno al 50%.

Dalla figura, si può notare che nella riga inerente il probe di Torpagnotta si ha un'occupazione della memoria Flash molto maggiore che negli altri siti, ciò è dovuto al fatto che in questo sito viene ancora utilizzata una delle prime versioni del BeagleBone Black che possiede una eMMC da 2GB rispetto a quella da 4GB dei nuovi modelli.

3.3.5 L'interfaccia "code_panel"

Sui BeagleBone e sul server è stato implementato anche un meccanismo per la registrazione e la visualizzazione di:

- anomalie di funzionamento;
- interventi sui siti;
- altre informazioni utili per la manutenzione.

Quando sui uPC vengono rilevati dei problemi, questi, automaticamente, vengono codificati ed inviati al server. Ad esempio, in caso di riavvio del sistema, sul device viene eseguito il comando:

curl --silent "http://ip.server/site_code.php?sito=site_name&code=1"

questo, richiama la pagina PHP "site_code", sul server all'indirizzo "ip.server", e gli passa il parametro identificativo del sito "site_name" ed il codice "1" che equivale a "sistema riavviato".

La pagina "site_code.php" è una pagina nascosta, che è stata appositamente creata per ricevere questi codici dalle varie macchine dell'infrastruttura e memorizzarli in un database dedicato sul server.

Oltre a questa, è stata realizzata anche un'altra pagina (mista PHP e Javascript) che costruisce dinamicamente l'interfaccia "code_panel" mostrata in Figura 3.32.

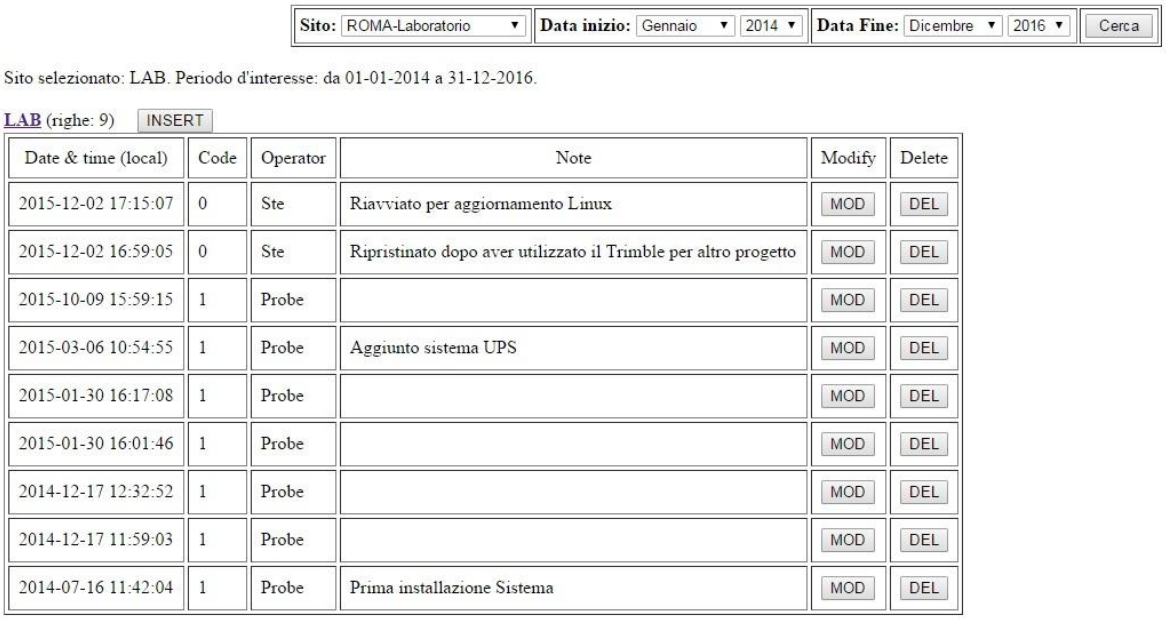


Figura 3.32 - Interfaccia web "code_panel".

In questo pannello, scegliendo il sito ed il periodo di interesse, viene costruita una tabella con tutte le informazioni memorizzate nel database di manutenzione da "site_code".

Qui un operatore è in grado di visualizzare la storia degli interventi e dei problemi su ogni sito, ad esempio, in figura viene mostrato tutto lo storico (dalla prima installazione ad oggi) del sistema di monitoraggio operante nel laboratorio (Roma - LAB).

Inoltre, da questo pannello, tramite i pulsanti "INSERT", "MOD" e "DEL" è possibile agire direttamente sul DB aggiungendo, modificando o eliminando delle righe, in modo che l'operatore possa aggiungere informazioni utili per la storia e la manutenzione del sistema. Queste operazioni sul database sono possibili ed enormemente agevolate da un'altra interfaccia che si apre una volta premuto uno dei pulsanti. Premendo "MOD" della prima riga in alto, ad esempio, si apre la schermata mostrata in Figura 3.33 in cui vengono riportati tutti i campi di quel record memorizzati nel DB, per la loro modifica.

Figura 3.33 - Popup visualizzato per la modifica di un record sul DB di manutenzione.

Ovviamente, per effettuare delle modifiche, è stato inserito un sistema di autenticazione che, una volta premuto il tasto "Update DB", richiede all'operatore delle credenziali prima di fare qualunque modifica (Figura 3.34).

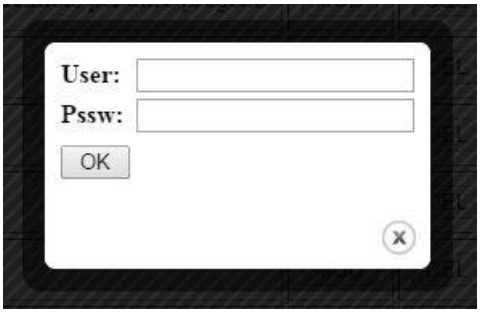


Figura 3.34 - Sistema di autenticazione per modificare il DB di manutenzione.

Questa interfaccia, oltre a fornire la "storia" del sito, utilissima durante le attività di manutenzione, ha anche il grande vantaggio di essere completamente accessibile da browser, il che permette di poterne usufruire da ovunque e quindi, anche da uno smartphone proprio durante gli interventi di riparazione delle apparecchiature nei siti.

Conclusioni

L'obiettivo di questo lavoro è stato quello di sviluppare un'infrastruttura adatta al monitoraggio geografico della Power Quality in bassa tensione su **larga scala**, ovvero un'infrastruttura che possa avere numerosi punti di osservazione (anche centinaia o più) distribuiti ovunque sul territorio e che abbia la capacità di gestirli e controllarli tutti da remoto.

Con questa visione, quindi, lo scopo è stato quello di sviluppare anche un'infrastruttura che fosse "**completa**", ecco che quindi ci si è occupati non solo dell'hardware per acquisire i segnali fisici dalla rete elettrica e del software per elaborarli, ma si è progettato e sviluppato anche tutto un meccanismo per poter gestire facilmente l'intero sistema, costituito da un server centrale e numerosi probe. Questo perché, sicuramente in un apparato di monitoraggio la scheda di acquisizione è un componente molto importante, ma in un sistema *distribuito geograficamente su larga scala*, anche i metodi utilizzati per la centralizzazione e la visualizzazione dei dati sono egualmente importanti. Inoltre, a mio avviso, in una struttura del genere, assume un'importanza ancora maggiore *un meccanismo che*, a livello superiore, *riesca a gestire e controllare da remoto i vari sistemi distribuiti sul territorio*, permettendo rapidamente la manutenzione ed il ripristino dei probe in caso di anomalie, senza la necessità di raggiungere fisicamente i siti di monitoraggio, in maniera quindi molto più immediata, economica ed agevole e riducendo notevolmente i "tempi morti" nei dati acquisiti.

Come si è già detto nel secondo capitolo, attualmente in commercio esistono numerosi dispositivi per misurazioni di PQ ma hanno lo svantaggio di essere molto costosi e spesso poco configurabili, il che non li rende molto adatti ad un monitoraggio su centinaia di siti e per ambito scientifico.

Quindi, per raggiungere gli obiettivi di questo lavoro, si è scelto di sviluppare questa infrastruttura partendo dalla "vecchia" strumentazione realizzata ad hoc dal laboratorio di Misure Elettriche ed Elettroniche (descritta nel paragrafo 2.4.2.1), migliorandola

notevolmente e cercando di risolvere tutti i suoi punti critici per l'applicabilità in un progetto di questo tipo.

Le principali limitazioni del "vecchio" sistema di monitoraggio del MeaLab sono state ampiamente discusse all'inizio del capitolo 3, e possono essere brevemente riassunte come segue:

a) Acquisizioni con elevata incertezza sull'istante preciso di inizio campionamento, sia localmente sia rispetto a file con stesso timestamp provenienti da siti differenti (anche superiori al minuto);
b) Costo del singolo punto di monitoraggio (intorno ai 2000€);
c) Bassa affidabilità dei sistemi;
d) Dimensioni delle apparecchiature (circa 80 dm^3);
e) Consumi (intorno ai 30 W);
f) Autonomia di memorizzazione e sicurezza dei dati;
g) Assenza di un sistema per gestire l'intera infrastruttura di monitoraggio con tutti i suoi siti.

Dopo una lunga serie di step evolutivi, che hanno completamente stravolto il sistema di partenza e che man mano hanno migliorato e risolto varie problematiche, sia hardware che software, si è giunti alla versione finale illustrata nel capitolo precedente. Questa versione è attualmente operante nei siti di monitoraggio del MeaLab, distribuiti nel comune di Roma come mostrato in Figura 4.1. Qui si può vedere che sono state mantenute le vecchie postazioni presso le centrali di Telecom Italia, mostrate nel paragrafo 2.4.2.1 (sostituendo le preesistenti apparecchiature), ed, inoltre, è stata aggiunta una postazione all'interno dell'Università degli Studi Roma Tre (nel laboratorio di Misure Elettriche ed Elettroniche).

Figura 4.1 – Posizione dei siti operanti con la nuova infrastruttura nel territorio di Roma.

Questa nuova infrastruttura realizzata supera completamente i "vecchi" limiti e punti deboli, infatti, come si è visto, tra le sue caratteristiche presenta:

a) **Sincronizzazione automatica delle acquisizioni con segnale GPS**. Rispetto al vecchio sistema in cui era il PC a "chiedere" alla scheda di acquisire, ora è la scheda che autonomamente fa partire le acquisizioni, sincronizzandosi con un segnale GPS, e poi le invia al uPC. In questo modo, dato che il segnale GPS viene ricevuto praticamente nello stesso momento da tutte le schede, queste faranno partire le acquisizioni tutte contemporaneamente e con un'incertezza massima minore di 300 nS rispetto al PPS del GPS;

b) **Basso costo**. Come si può vedere in Tabella 4.1, si è ottenuto un risparmio del 70% sul costo del singolo punto di monitoraggio rispetto al vecchio sistema del

laboratorio. Nel calcolo del costo totale non sono state incluse le bobine di Rogowski ne gli integratori, in quanto, non sono variati rispetto al vecchio sistema e soprattutto ci possono essere situazioni in cui questi componenti non sono necessari perché potrebbe essere sufficiente monitorare solo la qualità della tensione, tralasciando quella della corrente. Inoltre, nei costi non sono stati considerati quelli del sistema operativo e dell'ambiente di sviluppo software, a pagamento per il vecchio sistema e completamente gratuiti per il nuovo;

Costi	**Vecchio sistema**	**Nuovo sistema**
Acquisition Board	300 €	380 €
PC / uPC	1100 €	50 €
Modem	30 €	30 €
Phone Controller	20 €	20 €
Sistema di alimentazione	250 €	60 €
Shelter esterno	140 €	50 €
Ventilazione	120 €	10 €
Totale	≈ 2000 €	≈ 600 €

Tabella 4.1 - Suddivisione dei costi tra i vari componenti di un probe.

c) **Alta affidabilità**. Con le nuove apparecchiature si è ridotto notevolmente il numero di interventi di manutenzione, inoltre, la maggior parte di quelli effettuati sono stati prontamente eseguiti da remoto;
d) **Ingombri ridotti**. I probe ora sono molto più compatti. Considerando di inserire tutto il probe in un contenitore metallico, mentre questo prima doveva avere una capacità di almeno 80 dm^3, adesso sono sufficienti appena 8 dm^3, con una riduzione, quindi, del 90% sugli ingombri;
e) **Consumi limitati**. Si è passati da un consumo per la vecchia struttura di circa 30 W, ad un consumo, per quella nuova, inferiore a 5 W, con un calo, quindi, dell'85%. Ciò ha permesso, a parità di autonomia energetica, di ridurre fortemente le dimensioni del sistema di alimentazione, contribuendo così alla riduzione di ingombri e pesi del probe;

f) **Alta autonomia di memorizzazione e sicurezza dei dati**. Avendo ora un sistema automatico per l'invio dei dati al server centrale, la memoria dei vari apparati viene continuamente liberata dei file, evitando così che possa saturarsi. Inoltre, il fatto di riunire tutti i file in una macchina (il server) permette facilmente di eseguire operazioni di backup periodiche sui dati di monitoraggio dell'intera infrastruttura;

g) **Presenza di un sistema per la gestione ed il controllo da remoto di tutte le macchine dell'infrastruttura**. L'implementazione di questo meccanismo di gestione e controllo all'interno dell'infrastruttura permette:
   - di avere, praticamente quasi in tempo reale, tutte le informazioni necessarie sullo stato di funzionamento dei vari apparati;
   - una manutenzione più immediata, economica ed agevole dei probe, senza la necessità di recarsi fisicamente nei siti di monitoraggio;
   - di prevenire il verificarsi di anomalie sui uPC, tenendo d'occhio i vari parametri;
   - il funzionamento continuo e per lungo tempo dell'intero sistema e quindi del monitoraggio.

Inoltre la nuova infrastruttura è anche:

h) **Modulare**. Per aggiungere un ulteriore punto di osservazione, oltre ovviamente all'installazione fisica dell'apparato nel nuovo sito, sono sufficienti poche e semplici operazioni sul server (come aggiungere delle tabelle nel DB e delle cartelle nell'Hard disk);

i) **Accessibile ed utilizzabile da qualsiasi device**. Sia tutta la parte di visualizzazione dei dati di monitoraggio, sia quella per la gestione e manutenzione delle varie macchine è basata sul Web, quindi, è accessibile ed utilizzabile da qualsiasi dispositivo dotato di una connessione ad internet ed un browser, indipendentemente dalla sua architettura o dal suo sistema operativo (quindi anche da un comune smartphone).

Tutte queste caratteristiche sono state sintetizzate in Tabella 4.2, dove si ha anche un confronto con il vecchio sistema.

Caratteristiche	Vecchio sistema	Nuova infrastruttura
Incertezza sull'istante di inizio campionamento	Anche superiore al minuto	< 300 ns
Costo del singolo probe	≈ 2000 €	≈ 600 € (-70%)
Affidabilità	Bassa	Elevata
Dimensioni	≈ 80 dm^3	≈ 8 dm^3 (-90%)
Consumi	≈ 30 W	< 5 W (-85%)
Autonomia di memorizzazione	Media	Elevata
Sicurezza dei dati	Bassa	Elevata
Accessibilità	---	Da qualsiasi dispositivo
Modularità	---	Presente
Sistema per gestire e manutenere da remoto l'intera infrastruttura	Limitata al solo controllo telefonico dei probe	Avanzata con gestione completa da remoto di tutta l'infrastruttura

Tabella 4.2 - Caratteristiche a confronto tra il vecchio sistema e la nuova infrastruttura realizzata.

Con queste proprietà, si può quindi dire che è stata sviluppata un'infrastruttura particolarmente adatta per un monitoraggio della qualità dell'energia elettrica distribuito geograficamente su larga scala.

Per quanto riguarda gli sviluppi futuri di questo progetto, si sta già lavorando su diversi fronti, tra i quali quello di implementare una scheda che si integri con il BeagleBone Black e che quindi riesca a lavorare più a stretto contatto con questo uPC, sfruttandone appieno le sue potenzialità e fornendo nel complesso un dispositivo di acquisizione ancora più configurabile e con diversi gradi di libertà. Con la previsione di ridurre ulteriormente le dimensioni del probe del 3-4%, lasciando inalterato il costo finale e soprattutto mantenendo la stessa struttura di gestione e controllo sviluppata in questo lavoro.

Bibliografia

[Andersson] – T. Andersson, D. Nillson, "*Test and evaluation of voltage dip immunity*", Novembre 2002.

[Baggini] – A. Baggini, "*Handbook of Power Quality*", Wiley 2008.

[Balcells] – J. Balcells, V. Parisi, D. Gonzalez, "*New trends in power quality measuring instruments*", 2002.

[BeagleBone] – BeagleBone Black, http://beagleboard.org/black.

[BeagleBone2] – BeagleBone Black, http://elinux.org/Beagleboard:BeagleBoneBlack.

[BeagleBone3] – BeagleBone Black, http://www.ti.com/tool/beaglebk.

[Bhattacharyya] – Sharmistha Bhattacharyya, Sjef Cobben, "*Consequences of Poor Power Quality - An Overview*", in "Power Quality", InTech 2011.

[Bollen] – M.H. Bollen, I. Gu, "*Signal Processing of Power Quality Disturbances*", Wiley-IEEE Press 2006.

[Bollen2] – M.H. Bollen, "*Understanding Power Quality Problems: Voltage Sags and Interruptions*", Wiley-IEEE Press 1999.

[Caciotta] – M. Caciotta, S. Giarnetti, F. Leccese, D. Trinca, "*Development of an USB Data Acquisition System for Power Quality and Smart Metering applications*", 11th International Conference on Environment and Electrical Engineering (EEEIC), Maggio 2012, Venezia.

[Caciotta2] – M. Caciotta, F. Leccese, S. Giarnetti, S. Di Pasquale, "*Geographical monitoring of Electrical Energy Quality determination: The problems of the sensors*", 7th International Conference on Sensing Technology (ICST), Dicembre 2013, Wellington.

[Caciotta3] – M. Caciotta, S. Giarnetti, F. Leccese, D. Trinca, "*A multi-platform Data Acquisition Device for Power Quality metrological certification*", 9th International

Conference on Environment and Electrical Engineering (EEEIC), Maggio 2010, Praga – Repubblica Ceca.

[Caciotta4] – M. Caciotta, S. Giarnetti, G. Lattanzi Cinquegrani, F. Leccese, D. Trinca, "*Development and characterization of a multi-platform Data Acquisition System for Power Quality metrological certification*", International Conference on Renewable Energies and Power Quality (ICREPQ), Aprile 2011, Las Palmas de Gran Canaria - Spagna.

[CEI] – *CEI EN 50160:2008-04*, Norma Tecnica, Milano: CEI.

[Chattopadhyay] – S. Chattopadhyay, M. Mitra, S. Sengupta, "*Electric Power Quality*", Springer 2011.

[Dugan] – R.C. Dugan, M.F. McGranaghan, S. Santoso, H.W. Beaty, "*Electrical Power Systems Quality*", 2nd ed. McGraw-Hill 2003.

[Fuchs] – E.F. Fuchs, M.A.S. Masoum, "*Power Quality in Power Systems and Electrical Machines*", Elsevier 2008.

[GCC] – GNU Compiler Collection, http://www.gnu.org/software/gcc.

[Giarnetti] – S. Giarnetti, F. Leccese, M. Caciotta, "*Non recursive multi-harmonic least squares fitting for grid frequency estimation*", in "Measurement" vol. 66, Elsevier 2015.

[Herrera] – R.S. Herrera, A. Perez, P. Salmeron, J.R. Vazquez, S.P. Litran, "*Distortion Sources Identification in Electric Power Systems*", 2005.

[IEC] – *IEC 61000-4-30, Electromagnetic compatibility (EMC) - Part 4-30: Testing and measurement techniques - Power quality measurement methods*, 2003.

[IEEE] – *IEEE 1159-2009, Recommended Practice for Monitoring Electric Power Quality*, 2009.

[IEEE2] – *IEEE 100-1996, The IEEE Standard Dictionary of Electrical and Electronics Terms Sixth Edition*, 1996.

[Manson] – J. Manson, R. Targosz, "*European power quality survey report*", Leonardo Energy, Novembre 2008, http://www.leonardo-energy.org.

[MAX1320] – *MAX1320 ADC Data Sheet*, http://www.maximintegrated.com.

[McGranaghan] – M. McGranaghan, "*Monitoring morphs into customer service quid pro quo*", in "Electric Light & Power" vol. 79 n. 6, 2001.

[Resolution-T] – *Resolution-T GPS Timing Receiver Data Sheet*, http://www.trimble.com.

[Samotyj] – M. Samotyj, "*The Cost of Power Disturbances to Industrial & Digital Economy Companies*", 2001.

[Schipman] – Dr. Kurt Schipman, Dr. François Delincé, "*The importance of good power quality*", 2010.

[Shtargot] – J. Shtargot, "*Advanced Power-Line Monitoring Requires a High-Performance, Simultaneous-Sampling ADC*", in "Maxim Integrated Application Note 4281", Settembre 2008.

[Sumper] – A. Sumper, S. Galceran-Arellano, "*Monitoring Power Quality*", in "Handbook of Power Quality", Wiley 2008.

[Targosz] – R. Targosz, J. Manson, "*PAN European LPQI power quality survey*", 19th International Conference on Electricity Distribution (CIRED), Maggio 2007, Vienna.

[Ubuntu] – S.O. Linux Ubuntu, http://www.ubuntu.com.

[Webmin] – Interfaccia di amministrazione Webmin, http://www.webmin.com.

Bibliografia dell'Autore

S. Di Pasquale, S. Giarnetti, F. Leccese, D. Trinca, M. Cagnetti, M. Caciotta: "*A Distributed Web-Based System for Temporal and Spatial Power Quality Analysis*", capitolo di "Power Quality", Ottobre 2015, InTech - open science, open minds.

S. Di Pasquale, S. Giarnetti, F. Leccese, D. Trinca, M. Caciotta: "*A New Platform for High Accuracy Power Quality Measurements: the Forensic Point of View*", 20th IMEKO TC-4 International Symposium Measurement of Electrical Quantities, 15-17 Settembre 2014, Benevento - Italia.

S. Giarnetti, F. Leccese, S. Di Pasquale, M. Caciotta: "*Stima non ricorsiva della frequenza fondamentale della tensione di rete mediante fitting multiarmonico*", XXXI Congresso Nazionale GMEE, 11-13 Settembre 2014, Ancona - Italia.

S. Di Pasquale, S. Giarnetti, F. Leccese, M. Caciotta, D. Trinca, M. Cagnetti: "*Nuovo sistema per il monitoraggio geografico della Power Quality*", XXXI Congresso Nazionale GMEE, 11-13 Settembre 2014, Ancona - Italia.

F. Leccese, S. Giarnetti, D. Trinca, M. Cagnetti, S. Di Pasquale, M. Caciotta: "*Development and Characterization of a Multi-Platform Data Acquisition System for Power Quality Metrological Certification*", capitolo 6 di "Power Quality", Cambridge Scholars Publishing, 2014.

M. Caciotta, S. Di Pasquale, S. Giarnetti, F. Leccese, D. Trinca: "*A New Multi-Platform Data Acquisition System for Power Quality Metrological Certification*", "Journal of Energy and Power Engineering" vol. 8 n. 7, Luglio 2014, David Publishing Company, USA.

F. Leccese, M. Cagnetti, A. Calogero, D. Trinca, S. Di Pasquale, S. Giarnetti, L. Cozzella: "*A New Acquisition and Imaging System for Environmental Measurements: an Experience on the Italian Cultural Heritage*", Sensors (Basel) Special Issue on "Sensors for Cultural Heritage Diagnostics", 23 Maggio 2014.

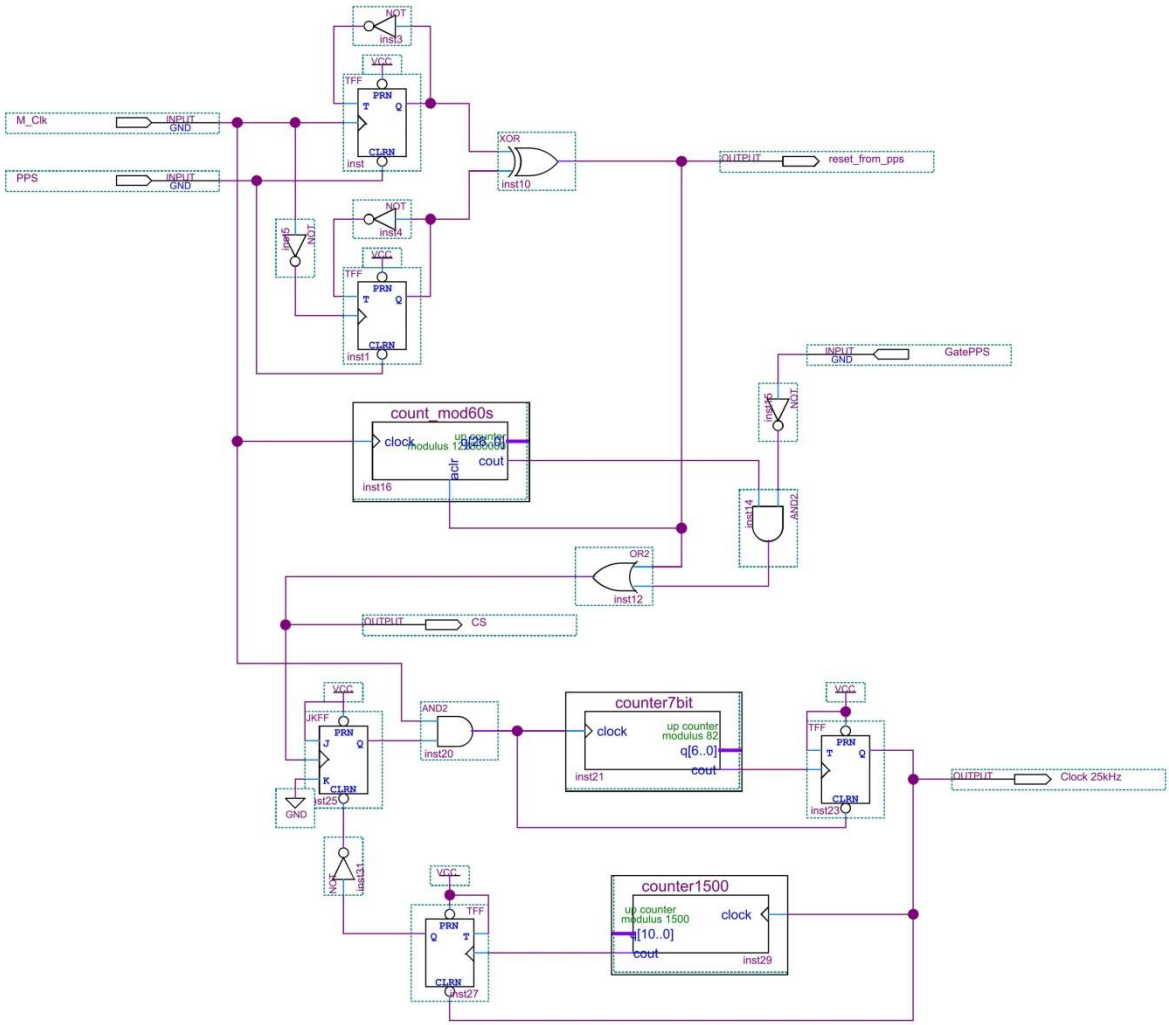
M. Caciotta, F. Leccese, S. Giarnetti, S. Di Pasquale: "*Geographical monitoring of Electrical Energy Quality determination: the problems of the sensors*", "Journal of Communication and Computer" vol. 10 n. 12, Dicembre 2013, David Publishing Company, USA.

Appendice

In questa sezione sono stati inseriti alcuni schemi sviluppati, il codice di alcune applicazioni realizzate ed altro materiale importante per lo sviluppo del progetto e per la comprensione del testo.

A1. Schema CPLD

Circuito logico principale implementato all'interno della CPLD (illustrato al Paragrafo 3.1.1).



A2. Procedura di preparazione del BeagleBone Black

Viene qui mostrata la procedura "passo-passo" per la preparazione del BeagleBone Black, con l'installazione del Sistema Operativo e di tutti i componenti necessari per il buon funzionamento dell'apparato (discusso al Paragrafo 3.2.1.1).

Preparazione uSD con Ubuntu da flashare su eMMC

Andare qui e scaricare l'immagine da flashare:

http://elinux.org/Beagleboard:Ubuntu_On_BeagleBone_Black

oppure qui:

http://elinux.org/BeagleBoardUbuntu#eMMC:_BeagleBone_Black

il link del file da scaricare (al momento di questa scrittura) è:

https://rcn-ee.net/deb/flasher/trusty/BBB-eMMC-flasher-ubuntu-14.04-console-armhf-2014-07-06-2gb.img.xz

(Controllare eventualmente nuove versioni dell'immagine sulle pagine indicate prima).

Per scaricare la iso (su Linux) si può anche eseguire il comando:

- wget https://rcn-ee.net/deb/flasher/trusty/BBB-eMMC-flasher-ubuntu-14.04-console-armhf-2014-07-06-2gb.img.xz

Per decomprimere la iso e poi per copiarla su SD:

- unxz BBB-eMMC-flasher-ubuntu-14.04-console-armhf-2014-07-06-2gb.img.xz
- sudo dd if=./BBB-eMMC-flasher-ubuntu-14.04-console-armhf-2014-07-06-2gb.img of=/dev/sdX

dove "/dev/sdX" indica proprio la memoria SD.

Dopo aver effettuato la copia sulla SD la si può espellere dal PC e sarà pronta per flashare l'eMMC del BeagleBone Black.

Eventualmente per eseguire la copia dell'immagine (già decompressa) su SD, in Windows, si può usare il programma "Win32DiskImager".

Preparazione completa del BeagleBone Black:

a) Inserire uSD nel BeagleBone e flashare eMMC:

- A BeagleBone spento: Inserire uSD nel uPC (inserire anche tastiera, monitor e cavo lan), premere e tenere premuto il pulsante "user" o "button" ed accendere il dispositivo inserendo la spina dell'alimentatore. Tenere premuto il pulsante (circa 5-10 secondi) finchè i 4 led non cominciano a lampeggiare in sequenza, poi rilasciare il tasto ed aspettare.
- Quando la copia su eMMC è finita (circa 6-7minuti) i led rimangono tutti accesi fissi, a questo punto disalimentare il BeagleBone, togliere uSD e riaccenderlo.

b) Configurazione della rete:

Cambiare indirizzo ip modificando il file "/etc/network/interfaces" tramite il programma nano:

sudo nano /etc/network/interfaces

Fare più versione del file "interfaces" (interfaces, interfaces2, interfaces3): una con DHCP, una con IP fisso per la rete del laboratorio ed una con IP fisso 192.168.1.40 (IP finale per i punti di monitoraggio).

Questi sono i parametri che bisogna avere per il DHCP:

```
# The primary network interface
auto eth0
iface eth0 inet dhcp
```

Questi i parametri che bisogna inserire per la rete del laboratorio:

```
# The primary network interface
auto eth0
iface eth0 inet static
address 192.168.XX.XX
netmask 255.255.255.0
network 192.168.XX.0
gateway 192.168.XX.1
```

Questi i parametri finali con IP fisso:

```
# The primary network interface
auto eth0
iface eth0 inet static
address 192.168.1.40
netmask 255.255.255.0
network 192.168.1.0
gateway 192.168.1.1
```

Cambiare DNS modificando il file "etc/resolvconf/resolv.conf.d/original" (sempre con nano):

sudo nano /etc/resolvconf/resolv.conf.d/original

ed aggiungere il DNS di Google (8.8.8.8), deve apparire così:

```
domain localdomain
search localdomain
nameserver 8.8.8.8
#nameserver 192.168.1.1
```

c) Aggiornamento del sistema.

Usare i comandi (anche più volte):

sudo apt-get update

sudo apt-get upgrade

sudo apt-get install -f

sudo apt-get dist-upgrade

Dopo gli aggiornamenti riavviare il sistema.

d) Impostazione lingua del sistema in Italiano

- Modificare il file "/etc/default/locale":

  sudo nano /etc/default/locale

  Modificare la lingua:

  LANG=it_IT.UTF-8

  LANGUAGE=it_IT:it

- Modificare anche il file "/var/lib/locales/supported.d/local":

  sudo nano /var/lib/locales/supported.d/local

  Mettendo sempre la lingua italiana.

- Eseguire i comandi:

  localedef --list-archive

  sudo locale-gen --purge it_IT.UTF-8 (cancella le altre lingue e lascia solo l'italiano)

  sudo dpkg-reconfigure locales

  locale -a

- Eseguite le modifiche riavviare (anche più volte) il sistema. Se si verificano dei problemi ripetere alcuni dei comandi visti sopra.

e) Impostazione layout tastiera italiano:

Eseguire il comando:

sudo dpkg-reconfigure keyboard-configuration

e selezionare la tastiera italiana.

Oppure, in alternativa, modificare il file "/etc/default/keyboard":

sudo nano /etc/default/keyboard

f) Installazione Webmin:

- Scaricare il pacchetto da installare con wget (ad esempio):

wget http://prdownloads.sourceforge.net/webadmin/webmin_1.700_all.deb

(Verificare sul sito di webmin che sia l'ultima versione).

- Installare il pacchetto con il comando:

  sudo dpkg --install webmin_1.700_all.deb

- Se l'installazione non viene completata con successo (molto probabile) installare anche tutte le dipendenze con il comando:

  sudo apt-get install perl libnet-ssleay-perl openssl libauthen-pam-perl libpam-runtime libio-pty-perl apt-show-versions python

- Poi correggere l'istallazione eseguendo il comando:

  sudo apt-get -f install

- Se da errore "apt-show-versions", bisogna togliere la compressione dei pacchetti dal sistema cancellando il file "/etc/apt/apt.conf.d/02compress-indexes" e poi aggiornando il sistema con i comandi:

  sudo rm /etc/apt/apt.conf.d/02compress-indexes

  sudo apt-get update

  sudo apt-get -f install

  A questo punto webmin dovrebbe essere installato correttamente e si può rimuovere il file d'installazione con:

  sudo rm webmin_1.700_all.deb

g) Controllare lista utenti con accesso al BeagleBone:

- Accedere a Webmin del BeagleBone dal browser del PC (ad es: https://192.168.1.40:10000)
- Andare nella sezione "System -> Users and Groups"
- Aprire la scheda di ogni utente e controllare che non abbiano la possibilità di login (tranne l'utente "ubuntu"), togliere l'accesso anche a "root" se ce l'ha.

h) Installazione servizio NTP e server I.N.Ri.M.:

- Per installare il demone NTP eseguire:

  sudo apt-get install ntp

- Poi per inserire i server NTP dell'I.N.Ri.M. bisogna modificare il file "/etc/ntp.conf" con il comando:

  sudo nano /etc/ntp.conf

  e bisogna aggiungere le seguenti righe (verificare sul sito dell'I.N.Ri.M. "http://www.inrim.it/ntp/config_i.shtml" che non siano cambiati):

  server ntp1.inrim.it # server primario

  server ntp2.inrim.it # server primario

- Per visualizzare i server attualmente utilizzati dal servizio NTP:

  ntpq -p

i) Modifica della password di accesso:

Per modificare la password lo si può fare sia da riga di comando sia tramite l'interfaccia di webmin. Tramite webmin bisogna andare alla sezione "System → Change Passwords", selezionare l'utente "ubuntu" e cambiargli la password. Tramite riga di comando bisogna digitare "passwd ubuntu" e "passwd root" e seguire le indicazioni.

j) Installazione firewall “ufw” e apertura porte:

Eseguire i seguenti comandi per l'installazione del firewall, per abilitarlo, per l'apertura delle porte 22 (SSH) e 10000 (webmin) e per verificarne lo stato:

```
sudo apt-get install ufw
sudo ufw enable
sudo ufw allow 22
sudo ufw allow 10000
```

Per verificarne lo stato di funzionamento eseguire il comando:

```
sudo ufw status verbose
```

k) Impostazione del logging degli IP sul server e del log in caso di riavvio del sistema:

Prima bisogna installare il servizio “curl” (se non è già installato) con:

```
sudo apt-get install curl
```

Poi bisogna pianificare 2 operazioni “curl” nel cronjob aggiungengo nel cron le righe:

- @reboot curl --silent "http://ip.server/site_code.php?sito=site_name&code=1" #Avvio_code
- 0,10,20,30,40,50 * * * * curl --silent http://ip.server/ip_logger.php?sito=site_name #Connection to ip_logger

tramite il comando (per editare il cron):

```
crontab -e
```

La prima volta che si esegue il comando sopra potrebbe venire chiesto quale editor di testo si preferisce per editare il cron, scegliere l'editor “nano”.

Per visualizzare tutte le operazioni pianificate eseguire il comando:

```
crontab -l
```

l) Cancellare riga con user e password nella schermata di login:

Su alcune versioni di Ubuntu 14.04, dopo che il BeagleBone si è avviato, nella schermata di login ci potrebbe essere scritto oltre all'indirizzo IP della macchina anche il nome utente e la password, per cancellare la riga con “user” e “pssw” bisogna modificare il file “/etc/issue” con:

```
sudo nano /etc/issue
```

e cancellare la relativa riga.

m) Aggiornare di nuovo il sistema e tutti i suoi componenti:

Utilizzare prima i comandi per aggiornare il sistema e poi quelli per liberare la cache:

```
sudo apt-get update
sudo apt-get upgrade
sudo apt-get install -f
sudo apt-get dist-upgrade
sudo apt-get autoremove
```

sudo apt-get clean

Dopo gli aggiornamenti riavviare il BeagleBone.

n) Mettere IP finale (192.168.1.40):

Andare nella directory "/etc/network" e rinominare i file "interfaces" che sono stati preparati in precedenza seguendo i comandi:

cd /etc/network

Rinominare il file che attualmente si chiama "interfaces" in "interfacesX" e poi rinominare il file con IP 192.168.1.40 in "interfaces":

sudo mv interfaces interfaces4

sudo mv interfaces3 interfaces

o) Ora il BeagleBone Black è pronto!!

A3. Sorgente di "*configuration*"

Codice dell'applicativo per la configurazione iniziale della macchina, mostrato al Paragrafo 3.2.1.2.1 ed in Figura 3.14.

```
/*************************************************************************
 *
 *          Programma di configurazione/calibrazione della scheda
 *
 *************************************************************************/

#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>           // Needed for read
#include <fcntl.h>
#include <termios.h>
#include <string.h>           // Needed for memset
#include <time.h>
#include <sys/time.h>
#include <sys/stat.h>         // To manage directories
#include <errno.h>            // To print error messages
#include <stdarg.h>
#include <math.h>

#define     DEVICE_NAME "TEST"            // Nome della macchina
#define     DIR_DATA    "/data"           // Directory principale di
raccolta dati "data"
#define     config_name "/data/config.txt"      // File di configurazione
#define     MAX_DATA_LENGTH   24000
#define     MAX_TXT_LINE_LENGTH     100
#define     FALSE       0
#define     TRUE        1

/** Variabili globali **/
static const char *PORT_DATA = "/dev/ttyUSB0";       // Servizio per la
porta Data

int nport_data = -1;          // Contiene il numero della porta seriale
(se aperta)
struct termios options_data;  // Contiene i settaggi della porta Data

char read_data[MAX_DATA_LENGTH + 1] = {0};      // Array contenente tutti
i dati ricevuti da port "data"
char txt_data[MAX_TXT_LINE_LENGTH] = {0};       // Array usato per la
scrittura e la lettura del file txt (riga per riga)

/* Variabili globali per i timer di controllo tra pacchetti */
struct timeval current_time, last_packet_time;

/** Variabili Globali da calcolare ed inserire nel file config.txt **/
double offsetVR = 0;                  // Offset sulla fase R
double offsetVS = 0;                  // Offset sulla fase S
double offsetVT = 0;                  // Offset sulla fase T
double offsetVN = 0;                  // Offset sul neutro
double VconvR = 0.06;         // Fattore di conversione Tensione fase R
double VconvS = 0.06;         // Fattore di conversione Tensione fase S
double VconvT = 0.06;         // Fattore di conversione Tensione fase T
double VconvN = 0.06;         // Fattore di conversione Tensione del
```

```
Neutro

/** Lista delle funzioni **/
int open_data_port();
int open_ts_port();
void clean_port(int port_number);
void make_directory(const char *directory);
void offset_processing(void);
void config_save();
void config_read(void);


/******************  Programma principale  ************************/
int main(void) {

      /* Dichiarazioni ed inizializzazioni variabili */
      ssize_t chars_read_data = 0;  // Numero di caratteri letti da data
      int total_chars_data = 0;           // Numero totale di caratteri
letti da data
      char read_buffer_data[MAX_DATA_LENGTH + 1] = {0};     // Buffer
software di appoggio per la porta data
      int packets_data = 0;               // Numero di pacchetti ricevuti
da data

      int i;            // Variabile di servizio per ciclo for
      char done = 0;          // Indica se è stata effettuata o meno la
calibrazione (0=NO, 1=SI)
      char data_in[30] = {0}; // Contiene la stringa ricevuta da tastiera
(dall'utente)

      /** Inizio programma main **/
      /* Visualizza ed aspetta un messaggio di conferma per la
configurazione */
      printf("Sicuro di voler sovrascrivere il file \"config.txt\"?
[si/no/0]\n");
      gets(data_in);
      printf("Ricevuto: %s\n", data_in);

      /* Check sul valore ricevuto */
      if(strcmp(data_in, "si")==0) {            // La configurazione va
effettuata
            printf("Procedo con la configurazione!\n");

            /* Visualizza informazioni sul dispositivo */
            printf("\nDevice name: %s\n", DEVICE_NAME);          //
Visualizza il nome della macchina

            make_directory(DIR_DATA);           // Crea directory "data"
in root
            open_data_port();       // Apre la porta data ed elimina
tutti i dati nel buffer fisico

            // Carica il registro per il timer
            gettimeofday(&last_packet_time, NULL);          // per timer
tra pacchetti

            /* Loop principale */
            while(!done){

                  /* Ricezione da port "data" */
                  chars_read_data = read(nport_data, read_buffer_data,
MAX_DATA_LENGTH); // Legge dalla porta data
```

```
                if (chars_read_data > 0){          // Dati ricevuti

                        gettimeofday(&last_packet_time, NULL);          //
Carica timer tra pacchetti

                        // Controlla l'overflow del buffer software
                        if ((chars_read_data+total_chars_data) >
MAX_DATA_LENGTH) {
                                printf("Overflow del buffer software per la
ricezione di %d caratteri sulla porta data.\n",
chars_read_data+total_chars_data);

                        } else {    // Non c'è pericolo di overflow
                                for(i=0; i<chars_read_data; i++){
                                        read_data[i+total_chars_data] =
read_buffer_data[i];    // Accumula tutti i dati ricevuti
                                        read_buffer_data[i] = 0;      //
Resetta il buffer software di appoggio
                                }

                                packets_data++;         // conta i pacchetti
ricevuti
                                total_chars_data += chars_read_data;
      // conta i caratteri totali ricevuti

                                if (total_chars_data == MAX_DATA_LENGTH) {
      // Tutto il file data è stato ricevuto per intero
                                        total_chars_data = 0;         //
Resetta conteggio

                                        packets_data = 0; // Resetta conteggio
pacchetti

                                        offset_processing();          //
Calcola i valori degli offset

                                        printf("\nValori di offset calcolati:
\n");
                                        // Visualizza gli offset
                                        printf("VR: %.2f, VS: %.2f, VT: %.2f,
VN: %.2f;\n", offsetVR, offsetVS, offsetVT, offsetVN);
                                        printf("\nFattori di conversione:
\n");
                                        // Visualizza i fattori di
conversione
                                        printf("VR: %.2f, VS: %.2f, VT: %.2f,
VN: %.2f;\n", VconvR, VconvS, VconvT, VconvN);

                                        config_save();                // Salva
il file di calibrazione "config.txt"

                                        done = 1;         // File config
generato
                                }     // fine if==24000
                        }     // fine del check per eventuale overflow
                }           // fine ricezione dati da port data

                gettimeofday(&current_time, NULL);       // Carica
tempo attuale

                /* Controlla il tempo trascorso tra un pacchetto ed il
```

```
successivo */
                  if (total_chars_data!=0) {          // Controllo da
eseguire solo all'interno della ricezione dei 24000 byte
                        if ((current_time.tv_sec-last_packet_time.tv_sec)
> 3) {      // Sono passati più di 3 secondi
                              total_chars_data = 0;               //
Azzera conteggio caratteri "data" ricevuti
                              clean_port(nport_data); // Svuota il buffer
fisico della porta data
                              sleep(1);
                              printf("Valore porta chiusa: %d, numero
porta: %d.\n", (close(nport_data)), nport_data);
                              memset(&read_data, 0, sizeof(read_data));
      // Reset buffer software della porta data
                              sleep(1);
                              open_data_port();       // Apre la porta
data ed elimina tutti i dati nel buffer fisico
                              printf("Errore ricezione pacchetti.\n");
                        }
                  }
            }     // fine while(!done)

            close(nport_data);                  // Chiude la porta data
            config_read();                // Legge e visualizza i valori
scritti in config.txt
            printf("\nConfigurazione\\Calibrazione effettuata!!\n\n");

      } else if(strcmp(data_in, "no")==0) {          // La configurazione
non va eseguita
            printf("Configurazione non effettuata!!\n");
            printf("\nApro il file \"config.txt\" già esistente in sola
lettura.\n");
            config_read();                // Legge e visualizza i valori
scritti in config.txt

      } else if(strcmp(data_in, "0")==0) {           // La configurazione
va eseguita con gli offset a zero
            config_save();                // Salva il file "config.txt"
con tutti gli offset a zero
            config_read();                // Legge e visualizza i valori
scritti in config.txt
            printf("\nConfigurazione effettuata con tutti gli offset a 0
(zero)!!\n\n");

      } else {
            printf("Comando non riconosciuto!!\n");
      }

      return 0;
}     // fine main


/** Funzione che apre la porta seriale virtuale ttyUSB0 (DATA) **/
int open_data_port(){

      /* Apre la porta e ne restituisce il numero (o -1 se non la apre)
*/
      nport_data = open(PORT_DATA, O_RDWR | O_NOCTTY | O_NONBLOCK);    //
Apre in lettura e scrittura, non controllato da tty

      if (nport_data != -1){  // The serial port is open
            printf("Porta seriale aperta!\n");
```

```
            memset(&options_data,0,sizeof(options_data));  // Reset
options_data

            options_data.c_iflag = 0;     // Turn off input processing
(no parity, no flow control, ecc...)
            options_data.c_oflag = 0;     // Turn off output processing
(Raw output)
            /* 8n1, enable receiving characters, local connection(no
modem contol) */
            options_data.c_cflag = CS8 | CREAD | CLOCAL;
            options_data.c_lflag = IEXTEN;      // Enable FLUSH, No line
processing (non-canonical mode, no echoes)
            options_data.c_cc[VMIN] = 0;  // Minimun number of characters
to read
            options_data.c_cc[VTIME] = 5; // Inter-characters timeout (in
0.1sec)
            cfsetspeed(&options_data, B921600); // Communication speed
(sets Baud rate)

            tcsetattr(nport_data, TCSAFLUSH, &options_data); // Set
attributes immediately from options_data and any data received and not
read will be discarded
      } else {
            printf("Impossibile aprire %s\n", PORT_DATA);
      }
      return (nport_data);
}


/** Funzione che pulisce il buffer fisico della porta seriale
TIMESTAMP(nport_ts) o DATA(nport_data) **/
void clean_port(int port_number) {
      int buffer_deleted = -1;            // Indica se il buffer fisico
della seriale è stato svuotato o meno
      buffer_deleted = tcflush(port_number, TCIOFLUSH);          //
Svuota il buffer fisico della seriale (input&output)
      if (buffer_deleted != -1) {               // Visualizza se il buffer
è stato svuotato o meno
            printf("Buffer porta %d svuotato, valore: %i.\n",
port_number, buffer_deleted);
      } else {
            printf("Buffer porta %d non svuotato!\n", port_number);
      }
}


/** Funzione che crea una directory se non esiste e fa un check sulla
creazione **/
void make_directory(const char *directory) {
      int dir_fail = -1;            // Indica se la creazione delle
directories è andata a buon fine
      errno = 0;              // Resetta la variabile contenente l'errore

      dir_fail = mkdir(directory, 0755);        // Crea la directory
passata come argomento con permessi 755
      if(dir_fail == 0) {           // Creazione andata a buon fine
            printf("Directory \"%s\" creata.\n", directory);
      } else {                // Creazione non andata a buon fine
            printf("Directory \"%s\" ", directory);
            if (errno == EEXIST) {        // Directory già esistente
                  printf("già esistente.\n");
```

```
            } else {                        // Altro problema nella
creazione della directory
                  printf("non creata.\n");
            }
      }
}


/** Funzione che genera e salva il file "config.txt" **/
void config_save(void) {
      /* Dichiarazioni variabili */
      size_t file_elem_txt;         // Elementi scritti nel file txt
      FILE *txt_file;               // Handler del file txt

      /* Crea e riempie un file txt con i valori calcolati */
      txt_file = fopen(config_name, "w");       // Crea file "config.txt"

      // Scrivo il file txt provvisorio riga per riga usando un array di
appoggio
      fputs(DEVICE_NAME, txt_file);             // Prima riga col nome
della macchina/del sito

      sprintf(txt_data, "\n%.2f;%.2f;%.2f;%.2f;\n", offsetVR, offsetVS,
offsetVT, offsetVN);    // Seconda riga con gli offset
      fputs(txt_data, txt_file);          // Scrive una riga sul file

      sprintf(txt_data, "%.2f;%.2f;%.2f;%.2f;\n", VconvR, VconvS, VconvT,
VconvN);          // Terza riga con i fattori di conversione
      fputs(txt_data, txt_file);          // Scrive una riga sul file

      fclose(txt_file);       // Chiude lo stream verso il file
      printf("\nFile di configurazione generato in \"%s\"!!\n",
config_name);           // Visualizza messaggio di conferma
}


/* Funzione che legge le informazioni contenute nel config.txt (nome
device, offset e fattori di conversione) */
void config_read(void) {
      /* Dichiarazioni ed inizializzazioni variabili */
      FILE *config_file;                  // Handler del file config
      int i = 0;
      memset(&txt_data, 0, sizeof(txt_data));        // Reset buffer di
appoggio per il txt

      printf("Contenuto del file:\n");
      /* Apre il file config, legge e visualizza i valori */
      config_file = fopen(config_name, "r");               // Apre il
file di configurazione "config.txt"
      for(i=0; i<3; i++) {                // Legge e visualizza le 3
righe di config.txt
            fscanf(config_file, "%s\n", txt_data);         // Legge la
riga i-esima
            printf("%s\n", txt_data);
      }
      fclose(config_file);                // Chiude lo stream verso il
file config
}


/** Funzione che suddivide i campioni acquisiti e calcola i valori che
poi andranno nel file config.txt **/
```

```
void offset_processing(void) {
	/* Dichiarazione variabili */
	short LSB = 0;
	short MSB = 0;
	double campione = 0;
	short i = 0;
	short sample_num = 0;
	char channel_num = 0;
	double accVr = 0;
	double accVs = 0;
	double accVt = 0;
	double accVn = 0;
	double Vr[1500] = {0};
	double Vs[1500] = {0};
	double Vt[1500] = {0};
	double Vn[1500] = {0};

	for(i=0; i<MAX_DATA_LENGTH; i=i+2) {
		LSB = 0;		// Azzero le variabili
		MSB = 0;
		campione = 0;
		MSB = read_data[i] & 63;			// Carica l'MSB
nella variabile d'appoggio e fa il cast da char a short
		LSB = read_data[i+1];			// Carica l'LSB nella
variabile d'appoggio e fa il cast da char a short
		campione = (double) ((MSB<<8) + LSB);	// Ricostruisce il
campione

		if (campione>=8192) {		// Se negativo ne calcola il
complemento a 2
			campione -= 16384;
		}

		// Separa i campioni negli array delle tensioni e delle
correnti e dopo aggiunge l'offset
		switch (channel_num) {
			case 0:
				Vr[sample_num] = campione;
				break;
			case 1:
				//I_R[sample_num] = campione;
				break;
			case 2:
				Vs[sample_num] = campione;
				break;
			case 3:
				//I_S[sample_num] = campione;
				break;
			case 4:
				Vt[sample_num] = campione;
				break;
			case 5:
				//I_T[sample_num] = campione;
				break;
			case 6:
				Vn[sample_num] = campione;
				break;
			case 7:
				//I_N[sample_num] = campione;
				break;
		}
```

```
            channel_num++;

            if (channel_num>=8) {
                  channel_num = 0;
                  sample_num++;
            }
      }

      // Fa la media dei valori acquisiti sulle singole fasi e la carica
come offset
      for(i=0; i<=1499; i++) {
            accVr += Vr[i];
            accVs += Vs[i];
            accVt += Vt[i];
            accVn += Vn[i];
      }
      offsetVR = (accVr / 1500);
      offsetVS = (accVs / 1500);
      offsetVT = (accVt / 1500);
      offsetVN = (accVn / 1500);
}     // Fine offset_processing
```

A4. Sorgente di "*capture*"

Programma principale per l'acquisizione e l'elaborazione dei dati provenienti dall'acquisition board. Discusso al Paragrafo 3.2.1.2.2 e mostrato in Figura 3.16.

```
/*****************************************************************
 *
 *          Programma principale di acquisizione dati da USB (FTDI)
 *
 *****************************************************************/

#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>           // Needed for read and getpid
#include <fcntl.h>
#include <termios.h>
#include <string.h>           // Needed for memset
#include <signal.h>
#include <time.h>
#include <sys/time.h>
#include <sys/stat.h>         // To manage directories
#include <errno.h>            // To print error messages
#include <stdarg.h>
#include <math.h>
#include <sys/resource.h>     // Needed to use get/setpriority

#define     MAX_DATA_LENGTH   24000
#define     MAX_TS_LENGTH           7
#define     MAX_TXT_LINE_LENGTH     120
#define     FALSE       0
#define     TRUE        1
#define     DIR_BIN           "/data/bin" // Directory "bin" in "data"
#define     DIR_TXT           "/data/txt" // Directory "txt" in "data"
#define     DIR_DATA    "/data"           // Directory principale di
raccolta dati "data"
#define     TEMP_BIN    "/data/temp_bin"  // File "bin" provvisorio in
"data"
#define     TEMP_TXT    "/data/temp_txt"  // File "txt" provvisorio in
"data"
#define     CONFIG_NAME "/data/config.txt"      // File di configurazione
#define     REPORT_NAME "/home/ubuntu/report.txt"     // File di report

/** Variabili globali **/
static const char *PORT_DATA = "/dev/ttyUSB0";       // Servizio per la
porta Data
static const char *PORT_TS = "/dev/ttyUSB1";         // Servizio per la
porta Time Stamp
char DEVICE_NAME[16] = {0};         // Nome della macchina

int nport_data = -1;          // Contengono il numero della porta seriale
(se aperta)
int nport_ts = -1;
struct termios options_data;  // Contiene i settaggi della porta Data
struct termios options_ts;// Contiene i settaggi della porta TimeStamp


/* Variabili globali che contengono il Time Stamp in formato numerico */
int anno = 0;
```

```
char mese = 1;
char giorno = 1;
char ora = 0;
char minuto = 0;

char ts_string[21] = {0};     // Stringa che contiene il Time Stamp in
formato "stringa"
char bin_name[30] = {0};      // Stringa che contiene il nome del file
bin (completo di percorso assoluto)
char txt_name[30] = {0};      // Stringa che contiene il nome del file
txt (completo di percorso assoluto)

char read_data[MAX_DATA_LENGTH + 1] = {0};      // Array contenente tutti
i dati ricevuti da port "data"
char txt_data[MAX_TXT_LINE_LENGTH] = {0};// Array usato per la scrittura
del file txt (riga per riga)


/* Variabili globali per i timer di controllo (tra pacchetti e tra
blocchi dati) */
struct timeval current_time, last_packet_time, last_block_time;


/** Definizione delle costanti matematiche **/
#define     ROOT3       (sqrt(3))                // Radice quadrata di 3
#define     CONST1            (1./(2*ROOT3))     // Costante 1
#define     CONST2            (-1./6)            // Costante 2
#define     CONST3            (360./(2*M_PI))    // Costante 3
#define     n_periodi   3                  // Numero di periodi osservati
#define     CONST4            (n_periodi * M_PI)// Costante 4
#define     fc          (2048000. / 82)
#define     f0          (3. * (fc) / 1498)
#define     dt          (1./fc)
#define     w0          (2. * M_PI * f0)


/** Variabili Globali usate per il "txt_processing" **/
double alfaR[25], betaR[25];
double alfaS[25], betaS[25];
double alfaT[25], betaT[25];
double VoRe[25], VoIm[25], VdRe[25], VdIm[25], ViRe[25], ViIm[25];
double C0[25], S0[25];
double C1[25], S1[25];
double C2[25], S2[25];
double C3[25], S3[25];
double NVr[1500], NVs[1500], NVt[1500], NVn[1500];
double Vr[1500], Vs[1500], Vt[1500], Vn[1500];
double Vr_N[1500] = {0};
double Vs_N[1500] = {0};
double Vt_N[1500] = {0};
double Vr_eff, Vs_eff, Vt_eff, Vn_eff;
double THD = 0;
double THDR = 0;
double THDS = 0;
double THDT = 0;

double moduloR[25], moduloS[25], moduloT[25];
double moduloO[25], moduloD[25], moduloI[25];
double faseO[25], faseD[25], faseI[25];
double faseR[25], faseS[25], faseT[25];
double faseS_[25], faseT_[25];
double frequenza;
```

```
double offsetVR = 0;
double offsetVS = 0;
double offsetVT = 0;
double offsetVN = 0;
double VconvR = 1;            // Fattore di conversione Tensione
double VconvS = 1;            // Fattore di conversione Tensione
double VconvT = 1;            // Fattore di conversione Tensione
double VconvN = 1;            // Fattore di conversione Tensione


/** Lista delle funzioni **/
int open_data_port();
int open_ts_port();
void clean_port(int port_number);
void incr_min(void);
void genera_nomi_file(void);
void make_directory(const char *directory);
char report(const char *message, ...);
void save_files();
void channel_splitting(void);
void txt_processing();
double Fase4quadranti(double Re, double Im);
double ReComplexMul(double Re1, double Im1, double Re2, double Im2);
double ImComplexMul(double Re1, double Im1, double Re2, double Im2);
double CFA(double segnale[], char secondo_grado);
void config_loader(void);


/*********************  Programma principale  *********************/
int main(void) {
      /* Dichiarazione variabili necessarie e settaggio della priorità
massima per il processo */
      int nice_value;
      pid_t pid;

      pid = getpid();
      report("\nProcess ID: %d\n", pid);

      setpriority(PRIO_PROCESS, pid, -20);
      nice_value=getpriority(PRIO_PROCESS, pid);
      report("Nice value: %d\n", nice_value);

      /* Dichiarazioni ed inizializzazioni variabili */
      ssize_t chars_read_data = 0;  // Numero di caratteri letti da data
      int total_chars_data = 0;     // Numero totale di caratteri letti
da data
      char read_buffer_data[MAX_DATA_LENGTH + 1] = {0};    // Buffer
software di appoggio per la porta data
      int packets_data = 0;   // Numero di pacchetti ricevuti da data

      ssize_t chars_read_ts = 0;    // Numero di caratteri letti da ts
      int total_chars_ts = 0; // Numero totale di caratteri letti da ts
      char read_buffer_ts[MAX_TS_LENGTH + 1] = {0};  // Buffer software
di appoggio per la porta ts
      char read_ts[MAX_TS_LENGTH + 1] = {0};    // Array contenente tutti
i dati ricevuti da ts
      int packets_ts = 0;           // Numero di pacchetti ricevuti da ts

      int i;            // variabile di servizio per ciclo for
      char sync = FALSE;            // Indica se siamo sincronizzati o
meno col modulo gps
```

```
      char gps_sync = 3;           // Indica se il Trimble è agganciato
o meno al segnale gps (inizializzato in uno stato anomalo per farlo
entrare all'accensione in uno dei due if)
      char error_mess_written = FALSE;    // Indica se il messaggio
d'errore è stato già scritto o meno

      /** Inizio programma main **/
      config_loader();  // Carica le informazioni del dispositivo, gli
offset e i fattori di conversione

      /* Visualizza informazioni sul dispositivo */
      report("Device name: %s\n", DEVICE_NAME);      // Visualizza il
nome della macchina

      /* Crea le directories se non esistono già */
      make_directory(DIR_DATA);     // Crea directory "data" in root
      make_directory(DIR_BIN);      // Crea directory "bin" in "data"
      make_directory(DIR_TXT);      // Crea directory "txt" in "data"

      chdir(DIR_DATA);        // Prova ad andare nella directory "data"

      open_data_port();       // Apre la porta data ed elimina tutti i
dati nel buffer fisico
      open_ts_port();         // Apre la porta ts ed elimina tutti i dati
nel buffer fisico

      // Carica i registri per i timer
      gettimeofday(&last_packet_time, NULL);    // per timer tra pacchetti
      gettimeofday(&last_block_time, NULL);     // per timer tra blocchi
da 24000 byte


      /* Loop principale */
      while(1){

            /* Ricezione da port "data" */
            chars_read_data = read(nport_data, read_buffer_data,
MAX_DATA_LENGTH); // Legge dalla porta data

            if (chars_read_data > 0){          // Dati ricevuti

                  gettimeofday(&last_packet_time, NULL);    // Carica
timer tra pacchetti

                  // Controlla l'overflow del buffer software
                  if ((chars_read_data+total_chars_data) >
MAX_DATA_LENGTH) {
                        report("- Overflow del buffer software per la
ricezione di %d caratteri sulla porta data.\n",
chars_read_data+total_chars_data);

                  } else {    // Non c'è pericolo di overflow

                        for(i=0; i<chars_read_data; i++){
                              read_data[i+total_chars_data] =
read_buffer_data[i];    // Accumula tutti i dati ricevuti
                              read_buffer_data[i] = 0;      // Resetta il
buffer software di appoggio
                        }

                        packets_data++;   // conta i pacchetti ricevuti
                        total_chars_data += chars_read_data;     // conta
```

```
i caratteri totali ricevuti

                        if (total_chars_data == MAX_DATA_LENGTH) {//
Tutto il file data è stato ricevuto per intero
                              total_chars_data = 0;   // Resetta conteggio
                              incr_min();       // Incrementa minuto (il
ts arriva dopo il blocco data)
                              packets_data = 0; // Resetta conteggio
pacchetti
                              channel_splitting();    // Divide i campioni
acquisiti sui vari registri
                              txt_processing();       // Calcola i dati
che andranno nel file txt

                              if (sync == TRUE) {// Siamo sincronizzati
col modulo gps quindi salva i files

                                    genera_nomi_file();// Genera le
stringhe per i nomi dei file

                                    save_files();     // Genera e salva i
files "bin" e "txt"

                              } else {
                                    report("Non siamo sincronizzati col
modulo gps!\n");
                              }

                              if (minuto == 59) {     // Al 59°minuto
desincronizza il sistema in modo che venga
                                    sync = FALSE;     // risincronizzato
all'arrivo del time stamp al minuto 00
                              }

                              gettimeofday(&last_block_time, NULL);
      // Carica timer tra blocchi da 24000 byte
                        }     // fine if==24000
                  }     // fine del check per eventuale overflow
            }           // fine ricezione dati da port data


            /* Ricezione da port "ts" */
            chars_read_ts = read(nport_ts, read_buffer_ts,
MAX_TS_LENGTH);   // Legge dalla porta ts

            if (chars_read_ts > 0){       // Time Stamp was read.

                  for(i=0; i<chars_read_ts; i++){
                        read_ts[i+total_chars_ts] = read_buffer_ts[i]; //
Accumula tutti i dati ricevuti
                        read_buffer_ts[i] = 0;  // Resetta il buffer
software di appoggio
                  }

                  total_chars_ts += chars_read_ts;          // conta i
caratteri totali ricevuti

                  if (total_chars_ts == MAX_TS_LENGTH) {    // Il Time
Stamp è stato ricevuto per intero
                        total_chars_ts = 0;           // Resetta conteggio

                        if (read_ts[6] == 0) {  // Il Trimble è agganciato
```

```
al segnale gps
                              // Calcola anno ed assegna mese, giorno,
ora e minuto alle variabili globali
                              anno = read_ts[0]*256 + read_ts[1];
                              mese = read_ts[2];
                              giorno = read_ts[3];
                              ora = read_ts[4];
                              minuto = read_ts[5];
                              if (gps_sync != TRUE) { // Prima non era
agganciato
                                    report("Trimble agganciato al segnale
gps!\n");
                                    gps_sync = TRUE;  // Indico che ora è
agganciato
                              }
                        } else {                // Il Trimble non è
agganciato al segnale gps
                              if (gps_sync != FALSE) {// Prima era
agganciato
                                    report("Trimble NON agganciato al
segnale gps!\n");
                                    gps_sync = FALSE; // Indico che ora
non è agganciato
                              }
                        }

                        // Genera una stringa con il Time Stamp (per il
nome del file .bin)
                        sprintf(ts_string, "%.4d_%.2d_%.2d-%.2d_%.2d",
anno, mese, giorno, ora, minuto);
                        report("Time stamp ricevuto: %s\n", ts_string);//
Visualizza la stringa

                        if ((sync == FALSE) && (anno!=0)) { // Non era
sincronizzato col modulo gps quindi non aveva salvato i files (che
arrivano poco prima del ts)
// Se "anno!=0" vuol dire che almeno una volta il time stamp è stato
ricevuto dal modulo
                              report("Siamo nuovamente sincronizzati col
modulo gps.\n");

                              genera_nomi_file();// Genera le stringhe
per i nomi dei file

                              save_files();     // Genera e salva i files
"bin" e "txt"

                              sync = TRUE;      // Sistema sincronizzato
col modulo gps
                        }
                  }     // fine if == MAX_TS_LENGTH
            }           // fine ricezione Time Stamp

            gettimeofday(&current_time, NULL); // Carica tempo attuale

/* Controlla il tempo trascorso tra un pacchetto ed il successivo */
            if (total_chars_data!=0) {          // Controllo da eseguire
solo all'interno della ricezione dei 24000 byte
                  if ((current_time.tv_sec-last_packet_time.tv_sec) > 3)
{     // Sono passati più di 3 secondi
                        total_chars_data = 0;   // Azzera conteggio
caratteri "data" ricevuti
```

```
                        clean_port(nport_data); // Svuota il buffer fisico
della porta data
                        sleep(1);
                        report("Valore porta chiusa: %d, numero porta:
%d.\n", (close(nport_data)), nport_data);
                        memset(&read_data, 0, sizeof(read_data));      //
Reset buffer software della porta data
                        sleep(1);
                        open_data_port();       // Apre la porta data ed
elimina tutti i dati nel buffer fisico
                        report("- Errore ricezione pacchetti. %s\n",
ts_string);
                  }
            }

      /* Controlla il tempo trascorso tra un blocco da 24000 byte ed il
successivo */
            if ((current_time.tv_sec-last_block_time.tv_sec) > 63) {
                  if (error_mess_written == FALSE) {  // Per scrivere il
messaggio una sola volta
                        sync = FALSE;           // Abbiamo perso qualche
pacchetto da 24000 byte quindi aspettiamo il prossimo ts per
risincronizzarci
                        report("- Errore ricezione blocco da 24000.
%s\n", ts_string);
                        error_mess_written = TRUE;    // Messaggio scritto
                  }
            } else {
                  error_mess_written = FALSE;         // Messaggio di
errore eventualmente da riscrivere
            }
      }     // fine while(1)
      return 0;
}     // fine main


/** Funzione che apre la porta seriale virtuale ttyUSB0 (DATA) **/
int open_data_port(){
/* Apre la porta e ne restituisce il numero (o -1 se non la apre) */
      nport_data = open(PORT_DATA, O_RDWR | O_NOCTTY | O_NONBLOCK);    //
Apre in lettura e scrittura, non controllato da tty
      if (nport_data != -1){  // The serial port is open
            memset(&options_data,0,sizeof(options_data));  // Reset
options_data
            options_data.c_iflag = 0;     // Turn off input processing
(no parity, no flow control, ecc...)
            options_data.c_oflag = 0;     // Turn off output processing
(Raw output)
            /* 8n1, enable receiving characters, local connection(no
modem contol) */
            options_data.c_cflag = CS8 | CREAD | CLOCAL;
            options_data.c_lflag = IEXTEN;      // Enable FLUSH, No line
processing (non-canonical mode, no echoes)
            options_data.c_cc[VMIN] = 0;  // Minimun number of characters
to read
            options_data.c_cc[VTIME] = 5; // Inter-characters timeout (in
0.1sec)
            cfsetspeed(&options_data, B921600); // Communication speed
(sets Baud rate)

            tcsetattr(nport_data, TCSAFLUSH, &options_data); // Set
attributes immediately from options_data and any data received and not
```

```
read will be discarded
	} else {
		report("- Impossibile aprire %s\n", PORT_DATA);
		report("Chiudo il programma\n");
		exit(EXIT_FAILURE);		// Esce dal programma
	}
	return (nport_data);
}


/** Funzione che apre la porta seriale virtuale ttyUSB1 (TIME STAMP) **/
int open_ts_port(){
/* Apre la porta e ne restituisce il numero (o -1 se non la apre) */
	nport_ts = open(PORT_TS, O_RDWR | O_NOCTTY | O_NONBLOCK);  // Apre
in lettura e scrittura, non controllato da tty
	if (nport_ts != -1){			// The serial port is open
		memset(&options_ts,0,sizeof(options_ts));// Reset options_ts
		options_ts.c_iflag = INPCK;   // Enable input parity check
		options_ts.c_oflag = 0; // Turn off output processing (Raw
output)
		/* 8bits, enable receiver, parity enable, odd parity, local
connection(no modem contol) */
		options_ts.c_cflag = CS8 | CREAD | PARENB | PARODD | CLOCAL;
		options_ts.c_lflag = IEXTEN;  // Enable FLUSH, No line
processing (non-canonical mode, no echoes)
		options_ts.c_cc[VMIN] = 1;    // Minimun number of characters
to read
		options_ts.c_cc[VTIME] = 5;   // Inter-characters timeout (in
0.1sec)
		cfsetspeed(&options_ts, B9600);     // Communication speed
(sets Baud rate)

		tcsetattr(nport_ts, TCSAFLUSH, &options_ts);// Set attributes
immediately from options_ts and any data received and not read will be
discarded
	} else {
		report("- Impossibile aprire %s\n", PORT_TS);
		report("Chiudo il programma\n");
		exit(EXIT_FAILURE);     // Esce dal programma
	}
	return (nport_ts);
}


/** Funzione che pulisce il buffer fisico della porta seriale
TIMESTAMP(nport_ts) o DATA(nport_data) **/
void clean_port(int port_number) {
	int buffer_deleted = -1;		// Indica se il buffer fisico
della seriale è stato svuotato o meno
	buffer_deleted = tcflush(port_number, TCIOFLUSH);		//
Svuota il buffer fisico della seriale (input&output)
	if (buffer_deleted != -1) {		// Visualizza se il buffer è
stato svuotato o meno
		report("- Buffer porta %d svuotato, valore: %i.\n",
port_number, buffer_deleted);
	} else {
		report("- Buffer porta %d non svuotato!\n", port_number);
	}
}


/** Funzione che incrementa il minuto e di conseguenza la data tenendo
```

```
conto dei vari limiti sui valori ammissibili **/
void incr_min(void) {
      char fine_mese = 0;           // Numero dei giorni massimi
contenuti nel mese corrente
      minuto++;               // Incrementa minuto
      if (minuto == 60) {
            minuto = 0;
            ora++;                  // Incrementa ora
            if (ora == 24) {
                  ora = 0;
                  giorno++;         // Incrementa il giorno
                  switch (mese) {         // Imposta il numero di giorni
massimi contenuti nel mese corrente
                        case 1:           // Gennaio
                              fine_mese = 31;
                              break;
                        case 2:           // Febbraio
                              if (anno%4==0) {  // L'anno corrente è
bisestile
                                    fine_mese = 29;
                              } else {
                                    fine_mese = 28;
                              }
                              break;
                        case 3:           // Marzo
                              fine_mese = 31;
                              break;
                        case 4:           // Aprile
                              fine_mese = 30;
                              break;
                        case 5:           // Maggio
                              fine_mese = 31;
                              break;
                        case 6:           // Giugno
                              fine_mese = 30;
                              break;
                        case 7:           // Luglio
                              fine_mese = 31;
                              break;
                        case 8:           // Agosto
                              fine_mese = 31;
                              break;
                        case 9:           // Settembre
                              fine_mese = 30;
                              break;
                        case 10:          // Ottobre
                              fine_mese = 31;
                              break;
                        case 11:          // Novembre
                              fine_mese = 30;
                              break;
                        case 12:          // Dicembre
                              fine_mese = 31;
                              break;
                  }
                  if (giorno > fine_mese) {
                        giorno = 1;
                        mese++;
                        if (mese == 13) {
                              mese = 1;
                              anno++;
                        }     // fine if mese
```

```
                }     // fine if giorno
            }     // fine if ora
      }     // fine if minuto
}


/* Funzione che carica le informazioni del dispositivo, gli offset e i
fattori di conversione */
void config_loader(void) {
      /* Dichiarazioni ed inizializzazioni variabili */
      FILE *config_file;                  // Handler del file config
      int i = 0;
      /* Apre il file config e carica i valori */
      config_file = fopen(CONFIG_NAME, "r");                // Apre il
file di configurazione "config.txt"
      i = fscanf (config_file, "%s\n", DEVICE_NAME);        // Carica
nell'eseguibile il nome del dispositivo
      i = fscanf (config_file, "%lf;%lf;%lf;%lf;\n", &offsetVR,
&offsetVS, &offsetVT, &offsetVN);         // Carica i valori degli offset
      i = fscanf(config_file, "%lf;%lf;%lf;%lf;", &VconvR, &VconvS,
&VconvT, &VconvN);      // Carica i valori dei rapporti di conversione
      fclose(config_file);                // Chiude lo stream verso il
file config
}


/** Funzione che genera le stringhe per i nomi dei files txt e bin **/
void genera_nomi_file(void) {
      // Genera una stringa con il Time Stamp nel formato desiderato
      sprintf(ts_string, "%.4d_%.2d_%.2d-%.2d_%.2d", anno, mese, giorno,
ora, minuto);
      sprintf(txt_name, "%s/%s.txt", DIR_TXT, ts_string);  // Genera
stringa completa per file txt
      sprintf(bin_name, "%s/%s.bin", DIR_BIN, ts_string);  // Genera
stringa completa per file bin
}


/** Funzione che crea una directory se non esiste e fa un check sulla
creazione **/
void make_directory(const char *directory) {
      int dir_fail = -1;            // Indica se la creazione delle
directories è andata a buon fine
      errno = 0;        // Resetta la variabile contenente l'errore
      dir_fail = mkdir(directory, 0755);       // Crea la directory
passata come argomento con permessi 755
      if (dir_fail == 0) {          // Creazione andata a buon fine
            report("Directory \"%s\" creata.\n", directory);
      } else {                // Creazione non andata a buon fine
            report("Directory \"%s\" ", directory);
            if (errno == EEXIST) {        // Directory già esistente
                  report("già esistente.\n");
            } else {                      // Altro problema nella
creazione della directory
                  report("non creata.\n");
            }
      }
}


/** Funzione che segnala eventuali anomalie o il normale funzionamento
del programma (sullo schermo ed in un file) **/
```

```
char report(const char *message, ...) {
	/* Dichiarazione variabili */
	va_list args;			// Lista degli argomenti (variabili)
passati in ingresso alla funzione
	char err_ret = -1;		// Variabile di uscita
	int string_elem;	// Contiene il numero di elementi che
compongono la stringa del messaggio
	char message_string[160] = {0};		// Contiene il messaggio
sottoforma di stringa
	FILE *report_file;		// Handler del file report
	va_start(args, message);	// Inizio lista argomenti
	string_elem = vsprintf(message_string, message, args);	//
Scrive il messaggio nella stringa "message_string"
	va_end(args);			// Fine argomenti

	/* Crea e riempie un file txt con il Report */
	report_file = fopen(REPORT_NAME, "a");		// Crea file di
report in "append mode"
	fputs_unlocked(message_string, report_file);	// Scrive sul file
(senza bloccarlo)
	fclose(report_file);		// Chiude lo stream verso il file di
report

	return (err_ret);
}


/** Funzione che genera e salva i files "bin" e "txt" nelle rispettive
directories **/
void save_files(void) {
	/* Dichiarazioni variabili */
	size_t file_elem_data;		// Elementi scritti nel file data
	size_t file_elem_txt;		// Elementi scritti nel file txt
	FILE *data_file;		// Handler del file data
	FILE *txt_file;			// Handler del file txt
	char corrotto = FALSE;		// Indica lo stato dei campioni
acquisiti (FALSE = file valido)
	short i = 0;// Variabile di servizio per la scansione dei valori

	/* Crea e riempie un file bin con i campioni acquisiti */
	data_file = fopen(TEMP_BIN, "wb");		// Crea file "bin"
provvisorio
	file_elem_data = fwrite(read_data, 1, sizeof(read_data),
data_file); // Scrive sul file
	fclose(data_file);		// Chiude lo stream verso il file
	rename(TEMP_BIN, bin_name);		// Sposta e rinomina
definitivamente il file bin nella directory "bin"
	/* Controlla se ci sono dei salti "anormali" sui valori assoluti
dei campioni acquisiti */
	for(i=1; i<=1497; i++) {
		if((fabs(Vr[i]-Vr[i-1]) > 80) || (fabs(Vs[i]-Vs[i-1]) > 80)
|| (fabs(Vt[i]-Vt[i-1]) > 80)) {
			corrotto = TRUE;  // I dati acquisiti presentano
anomalie
		}
	}
	/* Crea e riempie un file txt con i valori calcolati */
	txt_file = fopen(TEMP_TXT, "w");	// Crea file "txt" provvisorio

// Scrivo il file txt provvisorio riga per riga usando un array di
appoggio
	sprintf(txt_data, "%.3f;%.1f;%.1f;%.1f;%.1f;%.1f\n", frequenza,
```

```
Vr_eff, Vs_eff, Vt_eff, Vn_eff, THD);     // Prima riga
      fputs(txt_data, txt_file);          // Scrive una riga sul file

      for(i=1; i<=24; i++) {        // Le 24 righe successive con i
valori di tutte le armoniche
            sprintf(txt_data, "%.3f;%.3f;%.3f;%.3f;%.3f;%.3f\n",
moduloR[i], faseR[i], moduloS[i], faseS[i], moduloT[i], faseT[i]);
            fputs(txt_data, txt_file);          // Scrive una riga sul
file
      }
      if(corrotto == TRUE) {  // File con dati con anomalie
            fputs("corrotto", txt_file);
      } else {                // File con dati senza anomalie
            fputs("OK", txt_file);
      }
      fclose(txt_file);       // Chiude lo stream verso il file
      rename(TEMP_TXT, txt_name);               // Sposta e rinomina
definitivamente il file txt nella directory "txt"
}


/** Funzione che divide i campioni acquisiti nei rispettivi array di
tensioni e correnti **/
void channel_splitting(void) {
      /* Dichiarazione variabili */
      short LSB = 0;
      short MSB = 0;
      double campione = 0;
      short i = 0;
      short sample_num = 0;
      char channel_num = 0;

      for(i=0; i<MAX_DATA_LENGTH; i=i+2) {
            LSB = 0;                // Azzero le variabili
            MSB = 0;
            campione = 0;
            MSB = read_data[i] & 63;                  // Carica l'MSB
nella variabile d'appoggio e fa il cast da char a short
            LSB = read_data[i+1];               // Carica l'LSB nella
variabile d'appoggio e fa il cast da char a short
            campione = (double) ((MSB<<8) + LSB);     // Ricostruisce il
campione

            if (campione>=8192) {         // Se negativo ne calcola il
complemento a 2
                  campione -= 16384;
            }
            // Separa i campioni negli array delle tensioni e delle
correnti e aggiunge l'offset
            switch (channel_num) {
                  case 0:
                        NVr[sample_num] = campione;
                        Vr[sample_num] = (campione - offsetVR) * VconvR;
                        break;
                  case 1:
                        //I_R[sample_num] = campione;
                        break;
                  case 2:
                        NVs[sample_num] = campione;
                        Vs[sample_num] = (campione - offsetVS) * VconvS;
                        break;
                  case 3:
```

```
                        //I_S[sample_num] = campione;
                        break;
                case 4:
                        NVt[sample_num] = campione;
                        Vt[sample_num] = (campione - offsetVT) * VconvT;
                        break;
                case 5:
                        //I_T[sample_num] = campione;
                        break;
                case 6:
                        NVn[sample_num] = campione;
                        Vn[sample_num] = (campione - offsetVN) * VconvN;
                        break;
                case 7:
                        //I_N[sample_num] = campione;
                        break;
            }
            channel_num++;
            if (channel_num>=8) {
                channel_num = 0;
                sample_num++;
            }
        }
}


/** Funzione che calcola tutti i valori che poi andranno nel file txt **/
void txt_processing(void) {
        double fR = 0;
        double fS = 0;
        double fT = 0;
        double accVr = 0;
        double accVs = 0;
        double accVt = 0;
        double accVn = 0;
        int i = 0;

        for(i=0; i<=1499; i++) {
            Vr_N[i] = Vr[i] - Vn[i];
            Vs_N[i] = Vs[i] - Vn[i];
            Vt_N[i] = Vt[i] - Vn[i];
            accVr = accVr + pow(Vr[i], 2);
            accVs = accVs + pow(Vs[i], 2);
            accVt = accVt + pow(Vt[i], 2);
            accVn = accVn + pow(Vn[i], 2);
        }

        Vr_eff = sqrt(accVr / 1500);
        Vs_eff = sqrt(accVs / 1500);
        Vt_eff = sqrt(accVt / 1500);
        Vn_eff = sqrt(accVn / 1500);

        fR = CFA(Vr_N, TRUE);   // MODIFICA Vr_N
        for(i=1; i<=24; i++) {
            alfaR[i] = (2 * fR / n_periodi) * S0[i];
            betaR[i] = (2 * fR / n_periodi) * C0[i];
        }

        fS = CFA(Vs_N, TRUE);
        for(i=1; i<=24; i++) {
            alfaS[i] = (2 * fS / n_periodi) * S0[i];
            betaS[i] = (2 * fS / n_periodi) * C0[i];
```

```
	}

	fT = CFA(Vt_N, TRUE);
	for(i=1; i<=24; i++) {
		alfaT[i] = (2 * fT / n_periodi) * S0[i];
		betaT[i] = (2 * fT / n_periodi) * C0[i];
	}

	// Calcolo SEQUENZE
	for(i=1; i<=24; i++) {
		VoRe[i] = (alfaR[i] + alfaS[i] + alfaT[i]) / 3;
		VoIm[i] = (betaR[i] + betaS[i] + betaT[i]) / 3;
		VdRe[i] = alfaR[i]/3 + ReComplexMul(CONST2, CONST1, alfaS[i],
betaS[i]) + ReComplexMul(CONST2, -CONST1, alfaT[i], betaT[i]);
		VdIm[i] = betaR[i]/3 + ImComplexMul(CONST2, CONST1, alfaS[i],
betaS[i]) + ImComplexMul(CONST2, -CONST1, alfaT[i], betaT[i]);
		ViRe[i] = alfaR[i]/3 + ReComplexMul(CONST2, -CONST1,
alfaS[i], betaS[i]) + ReComplexMul(CONST2, CONST1, alfaT[i], betaT[i]);
		ViIm[i] = betaR[i]/3 + ImComplexMul(CONST2, -CONST1,
alfaS[i], betaS[i]) + ImComplexMul(CONST2, CONST1, alfaT[i], betaT[i]);
	}

	// CALCOLO MODULO-FASE
	for(i=1; i<=24; i++) {
		moduloR[i] = sqrt(pow(alfaR[i], 2) + pow(betaR[i], 2));
		faseR[i] = Fase4quadranti(alfaR[i], betaR[i]);
		moduloS[i] = sqrt(pow(alfaS[i], 2) + pow(betaS[i], 2));
		faseS[i] = Fase4quadranti(alfaS[i], betaS[i]);
		moduloT[i] = sqrt(pow(alfaT[i], 2) + pow(betaT[i], 2));
		faseT[i] = Fase4quadranti(alfaT[i], betaT[i]);
		faseS_[i] = (faseS[i] - faseR[i]) * CONST3;
		faseT_[i] = (faseT[i] - faseR[i]) * CONST3;

		moduloD[i] = sqrt(pow(VdRe[i], 2) + pow(VdIm[i], 2));
		faseD[i] = Fase4quadranti(VdRe[i], VdIm[i]) * CONST3;
		moduloO[i] = sqrt(pow(VoRe[i], 2) + pow(VoIm[i], 2));
		faseO[i] = (Fase4quadranti(VoRe[i], VoIm[i]) - faseD[i]) *
CONST3;
		moduloI[i] = sqrt(pow(ViRe[i], 2) + pow(ViIm[i], 2));
		faseI[i] = (Fase4quadranti(ViRe[i], ViIm[i]) - faseD[i]) *
CONST3;
	}

	THDR = 0;
	THDS = 0;
	THDT = 0;
	THD = 0;
	for(i=2; i<=24; i++) {
		THDR = THDR + moduloR[i];
		THDS = THDS + moduloS[i];
		THDT = THDT + moduloT[i];
		THD = THD + moduloD[i];
	}

	THDR = THDR * 100 / moduloR[1];
	THDS = THDS * 100 / moduloS[1];
	THDT = THDT * 100 / moduloT[1];
	THD = THD * 100 / moduloD[1];

	frequenza = (fR + fS + fT) / 3;
}	// Fine txt_processing
```

```
double Fase4quadranti(double Re, double Im) {
	double fase = 0;
	if(Re > 0) {
		fase = atan(-Im / Re);
	} else if(Re < 0) {
		fase = atan(-Im / Re) + M_PI;
	} else {
		if(Im > 0) {
			fase = M_PI_2;
		} else if(Im < 0) {
			fase = -1 * M_PI_2;
		} else {
			fase = 0;
		}
	}
	return fase;
}


double ReComplexMul(double Re1, double Im1, double Re2, double Im2) {
	return (Re1 * Re2 - Im1 * Im2);
}


double ImComplexMul(double Re1, double Im1, double Re2, double Im2) {
	return (Re1 * Im2 + Re2 * Im1);
}


double CFA(double segnale[], char secondo_grado) {
	int N = 1498;
	double t;
	double f;
	double AppCos;
	double AppSin;
	double Y, Y1;
	double LS = 0;
	double LS1 = 0;
	double LS2 = 0;
	double delta_omega;
	int i = 0;
	int k = 0;

	memset(&C0, 0, sizeof(C0));   // Reset delle variabili globali
	memset(&S0, 0, sizeof(S0));   // Reset delle variabili globali
	memset(&C1, 0, sizeof(C1));   // Reset delle variabili globali
	memset(&S1, 0, sizeof(S1));   // Reset delle variabili globali
	memset(&C2, 0, sizeof(C2));   // Reset delle variabili globali
	memset(&S2, 0, sizeof(S2));   // Reset delle variabili globali
	memset(&C3, 0, sizeof(C3));   // Reset delle variabili globali
	memset(&S3, 0, sizeof(S3));   // Reset delle variabili globali

	for(k=1; k<=24; k++) {
		for(i=0; i<=N; i++) {
			t = i * dt;
			AppCos = segnale[i] * cos(w0 * t * k);
			AppSin = segnale[i] * sin(w0 * t * k);

			S0[k] = S0[k] + AppSin * dt;
			S1[k] = S1[k] + AppSin * t * dt;
			S2[k] = S2[k] + AppSin * pow(t,2) * dt;
```

```
                  S3[k] = S3[k] + AppSin * pow(t,3) * dt;

                  C0[k] = C0[k] + AppCos * dt;
                  C1[k] = C1[k] + AppCos * t * dt;
                  C2[k] = C2[k] + AppCos * pow(t,2) * dt;
                  C3[k] = C3[k] + AppCos * pow(t,3) * dt;
            }
// -------- Correzione per calcolo integrali con la regola del trapezio -
----------
            S0[k] = S0[k] - segnale[N] * sin(w0 * N * dt * k) * dt / 2;
            S1[k] = S1[k] - segnale[N] * sin(w0 * N * dt * k) * N *
pow(dt,2) / 2;
            S2[k] = S2[k] - segnale[N] * sin(w0 * N * dt * k) * pow(N,2)
* pow(dt,3) / 2;
            S3[k] = S3[k] - segnale[N] * sin(w0 * N * dt * k) * pow(N,3)
* pow(dt,4) / 2;

            C0[k] = C0[k] - (segnale[0] + segnale[N] * cos(w0 * N * dt *
k)) * dt / 2;
            C1[k] = C1[k] - segnale[N] * cos(w0 * N * dt * k) * N *
pow(dt,2) / 2;
            C2[k] = C2[k] - segnale[N] * cos(w0 * N * dt * k) * pow(N,2)
* pow(dt,3) / 2;
            C3[k] = C3[k] - segnale[N] * cos(w0 * N * dt * k) * pow(N,3)
* pow(dt,4) / 2;
      }

      Y = segnale[N];
      Y1 = ((3. / 2) * segnale[N] - 2 * segnale[N - 1] + (1. / 2) *
segnale[N - 2]) / dt;

      LS = 0;
      LS1 = 0;
      LS2 = 0;
      for(k=1; k<=24; k++) {
            for(i=1; i<=24; i++) {
                  if(k==i) {
                        LS = LS + (2 * k * w0 * C0[k] * S1[k]) / CONST4 +
pow(C0[k],2) / CONST4 - (2 * k * w0 * C1[k] * S0[k]) / CONST4 -
pow(S0[k],2) / CONST4;
                        LS1 = LS1 - ((4 * Y * C0[k]) / pow(w0,2)) - (2 *
pow(k,2) * w0 * pow(C1[k],2)) / CONST4 + (2 * pow(k,2) * w0 * C0[k] *
C2[k]) / CONST4 + (8 * k * CONST4 * Y * S0[k]) / pow(w0,2) - (4 * k *
C1[k] * S0[k]) / CONST4 - (4 * k * Y * S1[k]) / w0 - (2 * pow(k,2) * w0 *
pow(S1[k],2)) / CONST4 + (2 * pow(k,2) * w0 * S0[k] * S2[k]) / CONST4;
                        LS2 = LS2 + (8 * CONST4 * pow(Y,2)) / pow(w0,4) +
(8 * Y * C0[k]) / pow(w0,3) - (16 * pow(k,2) * pow(CONST4,2) * Y * C0[k])
/ pow(w0,3) + (8 * CONST4 * Y1 * C0[k]) / pow(w0,4) + (24 * pow(k,2) *
CONST4 * Y * C1[k]) / pow(w0,2) - (6 * pow(k,2) * pow(C1[k],2)) / CONST4
- (8 * pow(k,2) * Y * C2[k]) / w0 + (2 * pow(k,2) * C0[k] * C2[k]) /
CONST4 - (16 * k * pow(CONST4,2) * Y1 * S0[k]) / pow(w0,4) + (2 *
pow(k,3) * w0 * C3[k] * S0[k]) / CONST4 + (8 * k * Y * S1[k]) / pow(w0,2)
+ (8 * k * CONST4 * Y1 * S1[k]) / pow(w0,3) - (6 * pow(k,3) * w0 * C2[k]
* S1[k]) / CONST4 - (2 * pow(k,2) * pow(S1[k],2)) / CONST4 + (6 *
pow(k,3) * w0 * C1[k] * S2[k]) / CONST4 + (6 * pow(k,2) * S0[k] * S2[k])
/ CONST4 - (2 * pow(k,3) * w0 * C0[k] * S3[k]) / CONST4;
                  } else {
                        LS = LS + (4 * pow(i,2) * C0[i] * C0[k]) /
((pow(i,2) - pow(k,2)) * CONST4);
                        LS1 = LS1 - ((8 * pow(i,2) * Y * C0[i]) /
((pow(i,2) - pow(k,2)) * pow(w0,2))) - (8 * pow(i,2) * Y * C0[k]) /
((pow(i,2) - pow(k,2)) * pow(w0,2)) - (4 * pow(i,3) * C0[k] * S1[i]) /
```

```
((pow(i,2) - pow(k,2)) * CONST4) - (4 * pow(i,2) * k * C0[i] * S1[k]) /
((pow(i,2) - pow(k,2)) * CONST4);
                        LS2 = LS2 + (32 * pow(i,2) * CONST4 * pow(Y,2)) /
((pow(i,2) - pow(k,2)) * pow(w0,4)) + (16 * pow(i,2) * Y * C0[i]) /
((pow(i,2) - pow(k,2)) * pow(w0,3)) + (16 * pow(i,2) * CONST4 * Y1 *
C0[i]) / ((pow(i,2) - pow(k,2)) * pow(w0,4)) + (16 * pow(i,2) * Y *
C0[k]) / ((pow(i,2) - pow(k,2)) * pow(w0,3)) + (16 * pow(i,2) * CONST4 *
Y1 * C0[k]) / ((pow(i,2) - pow(k,2)) * pow(w0,4)) - (4 * pow(i,4) * C0[k]
* C2[i]) / ((pow(i,2) - pow(k,2)) * CONST4) - (4 * pow(i,2) * pow(k,2) *
C0[i] * C2[k]) / ((pow(i,2) - pow(k,2)) * CONST4) + (16 * pow(i,3) * Y *
S1[i]) / ((pow(i,2) - pow(k,2)) * pow(w0,2)) + (16 * pow(i,2) * k * Y *
S1[k]) / ((pow(i,2) - pow(k,2)) * pow(w0,2)) + (8 * pow(i,3) * k * S1[i]
* S1[k]) / ((pow(i,2) - pow(k,2)) * CONST4);
                    }
                }
        }

        if(secondo_grado == FALSE) {
                delta_omega = -LS / LS1;
        } else {
                double root1, root2, delta;
                double c = LS;
                double b = LS1;
                double a = (LS2 / 2);
                delta = pow(b,2) - 4 * a * c;
                if(delta > 0) {
                        root1 = (-b + sqrt(delta)) / (2 * a);
                        root2 = (-b - sqrt(delta)) / (2 * a);
                        if(fabs(root1) <= fabs(root2)) {
                                delta_omega = root1;
                        } else {
                                delta_omega = root2;
                        }
                } else {
                        delta_omega = -b / (2 * a);
                }
        }
        f = (w0 + delta_omega) / (2 * M_PI);
        return f;
}                       // Fine CFA
```

A5. Sorgente di “*ftp_sender*”

Programma per la trasmissione dei file txt e bin al server (Paragrafo 3.2.1.2.3 e Figura 3.17).

```
/*****************************************************************
 *
 *    Programma per l'invio tramite FTP dei files nella cartella data
 *                            (bin e txt)
 *
 * Le directory "bin" e "txt" remote devono essere contenute in una
 * cartella che deve avere lo stesso nome della macchina locale
 * ("/data/DEVICE_NAME/bin" e "/data/DEVICE_NAME/txt"), mentre i files
 * in locale devono essere contenuti nelle directory "bin" e "txt"
 * dentro la cartella "data".
 *****************************************************************/

#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>           // needed for sleep()
#include <sys/types.h>
#include <dirent.h>
#include <stdarg.h>
#include <string.h>
#include <errno.h>            // To print error messages
#include <sys/stat.h>         // To manage directories
#include <curl/curl.h>

#define     HOST  "ftp://ip.server" // Parametri di connessione al server
#define     USER  "user"
#define     PASS  "password"

#define     FALSE       0
#define     TRUE        1
#define     LOCALE_BIN_DIR    "/data/bin"       // Directory "bin" locale
#define     LOCALE_TXT_DIR    "/data/txt"       // Directory "txt" locale
#define     CONFIG_NAME       "/data/config.txt"// File di configurazione
#define     REPORT_FILE_NAME  "/home/ubuntu/report_ftp_sender.txt"// Nome
(e percorso) del file di report


/** Variabili globali **/
char DEVICE_NAME[16] = {0};         // Nome della macchina
char REMOTE_BIN_DIR[60] = {0};      // URL directory "bin" sul server
char REMOTE_TXT_DIR[60] = {0};      // URL directory "txt" sul server
char remote_file_url[80] = {0};     // URL del file che si sta spedendo


/** Lista delle funzioni **/
char report(const char *message, ...);
int sel_files(const struct dirent *passed_file);
void device_name_loader(void);


/** Programma principale **/
int main(void) {
      /* Dichiarazioni ed inizializzazioni variabili */
      struct dirent **eps;    // Array di puntatori a strutture "dirent"
      int files_num = 0;      // Numero di files nella cartella
```

```
      int i, k;               // Variabile di servizio per la scansione
dei files e la ritrasmissione
      int removed = 0;        // Indica se il file è stato rimosso o meno

      FILE *file_handle;      // Handle del file che si sta spedendo
      CURL *curl_handle;      // Handle per il trasferimento FTP
      CURLcode result;// Contiene il codice d'errore della trasmissione

      device_name_loader();   // Carica il nome della macchina dal file
di configurazione

      /* Genera le stringhe per i nomi delle directory remote per i files
txt e bin */
      sprintf(REMOTE_BIN_DIR, "%s//data/%s/bin", HOST, DEVICE_NAME);
      // Genera stringa URL completa per directory remota "bin"
      sprintf(REMOTE_TXT_DIR, "%s//data/%s/txt", HOST, DEVICE_NAME);
      // Genera stringa URL completa per directory remota "txt"

      /* Visualizza informazioni sul dispositivo */
      report("\nDevice name: %s\n", DEVICE_NAME);// Visualizza il nome
della macchina

      curl_global_init(CURL_GLOBAL_ALL); // Inizializzazione "Globale"
di libcurl

/********** Spedizione del file "config.txt" sul server  **************/
      curl_handle = curl_easy_init();// Inizializzazione di curl (FTP)
      if(curl_handle == NULL) {
            report("Errore di inizializzazione libcurl!!\n");
      } else {    // Inizializzazione eseguita
            // Imposta il nome utente
            curl_easy_setopt(curl_handle, CURLOPT_USERNAME, USER);
            // Imposta la password
            curl_easy_setopt(curl_handle, CURLOPT_PASSWORD, PASS);
            // Abilita upload FTP
            curl_easy_setopt(curl_handle, CURLOPT_UPLOAD, 1L)
            // Usa la modalità di trasferimento "binary" (default)
            curl_easy_setopt(curl_handle, CURLOPT_TRANSFERTEXT, 0);
            // Apre il file in sola lettura
            file_handle = fopen(CONFIG_NAME, "rb");
            // Crea l'URL completo del file da spedire
            sprintf(remote_file_url, "%s//data/%s/config.txt", HOST,
DEVICE_NAME);
            // Imposta l'URL del file da spedire
            curl_easy_setopt(curl_handle, CURLOPT_URL, remote_file_url);
            // Imposta file locale da spedire
            curl_easy_setopt(curl_handle, CURLOPT_READDATA, file_handle);

            for (k=1; k<4; k++) {   // Fa 3 tentativi di trasmissione per
file in caso di fallimento
                  result = curl_easy_perform(curl_handle);// Trasmette il
file tramite FTP
                  if(result == CURLE_OK) {// File trasmesso con successo
                        fclose(file_handle);    // Chiude il file
                        k = 5;// Smette di trasmettere lo stesso file
                  } else {
                        if(k>=3) {  // Dopo 3 tentativi di trasmissione
falliti lo segnala
                              fclose(file_handle);    // Chiude il file
                              report("config.txt ");  // Fa il report col
nome del file
                              report("Transfer failed: %s\n",
```

```
curl_easy_strerror(result));
                        }
                  }
            }     // Chiude il for per la ritrasmissione
      }     // Chiude if curl!= NULL
      curl_easy_cleanup(curl_handle);// Chiude la connessione col server

/********** Spedizione dei file "bin" e "txt" sul server  *************/
      /* Loop principale */
      while(1){
            /* Controllo e trasferimento dalla directory "bin" */
            chdir(LOCALE_BIN_DIR);  // Entra nella directory "bin" locale

            // Scansiona la directory corrente e fornisce un array di
strutture "dirent" in ordine alfabetico e che soddisfano la funzione
"sel_files"
            files_num = scandir("./", &eps, sel_files, alphasort);

            if (files_num > 0) {    // Ci sono files "utili" nella
directory da spedire
                  report("Files da spedire: %d in %s.\n", files_num,
LOCALE_BIN_DIR);

                  curl_handle = curl_easy_init();     // Inizializzazione
di curl (FTP)
                  if(curl_handle == NULL) {
                        report("Errore di inizializzazione libcurl!!\n");

                  } else {    // Inizializzazione eseguita
                        curl_easy_setopt(curl_handle, CURLOPT_USERNAME,
USER);      // Imposta il nome utente
                        curl_easy_setopt(curl_handle, CURLOPT_PASSWORD,
PASS);      // Imposta la password
                        curl_easy_setopt(curl_handle, CURLOPT_UPLOAD,
1L);        // Abilita upload FTP
                        curl_easy_setopt(curl_handle,
CURLOPT_TRANSFERTEXT, 0);     // Usa la modalità di trasferimento
"binary" (default)

                        for (i=0; i<files_num; i++) { // Seleziona un file
alla volta
                              file_handle = fopen(eps[i]->d_name, "rb");
      // Apre il file in sola lettura
                              sprintf(remote_file_url, "%s/%s",
REMOTE_BIN_DIR, eps[i]->d_name);    // Crea l'URL completo del file da
spedire
                              curl_easy_setopt(curl_handle, CURLOPT_URL,
remote_file_url);       // Imposta l'URL del file da spedire
                              curl_easy_setopt(curl_handle,
CURLOPT_READDATA, file_handle);     // Imposta file locale da spedire


                              for (k=1; k<4; k++) {   // Fa 3 tentativi di
trasmissione per file in caso di fallimento
                                    result =
curl_easy_perform(curl_handle);     // Trasmette il file tramite FTP
                                    if(result == CURLE_OK) {// File
trasmesso con successo
                                          fclose(file_handle);// Chiude
il file
                                          removed = remove(eps[i]-
>d_name);         // Cancella il file in locale
```

```
                                        if(removed==0) {// File rimosso
                                              // report("%s Deleted\n",
eps[i]->d_name);
                                        } else {    // File non rimosso
                                              report("%s NOT Deleted\n",
eps[i]->d_name);
                                        }
                                        k = 5;// Smette di trasmettere
lo stesso file
                                  } else {
                                        if(k>=3) {  // Dopo 3 tentativi
di trasmissione falliti lo segnala
                                              fclose(file_handle);
      // Chiude il file
                                              report("%s ", eps[i]-
>d_name);         // Fa il report col nome del file
                                              report("Transfer failed:
%s\n", curl_easy_strerror(result));
                                        }
                                  }
                            }// Chiude il for per la ritrasmissione
                      }     // Chiude il for per la selezione dei files
                      // report("Trasferimento 'bin' completato!\n");
                }     // Chiude if curl!= NULL
                curl_easy_cleanup(curl_handle);     // Chiude la
connessione col server
          } else if (files_num == 0) {  // La cartella è vuota
                report("Directory %s vuota.\n", LOCALE_BIN_DIR);
          } else {          // Ci sono stati problemi o errori
                report("Impossibile aprire la directory %s.\n",
LOCALE_BIN_DIR);
          }

          /* Controllo e trasferimento dalla directory "txt" */
          chdir(LOCALE_TXT_DIR);  // Entra nella directory "txt" locale

          // Scansiona la directory corrente e fornisce un array di
strutture "dirent" in ordine alfabetico e che soddisfano la funzione
"sel_files"
          files_num = scandir ("./", &eps, sel_files, alphasort);

          if (files_num > 0) {          // Ci sono files "utili" nella
directory da spedire
                report("Files da spedire: %d in %s.\n", files_num,
LOCALE_TXT_DIR);

                curl_handle = curl_easy_init();     // Inizializzazione
di curl (FTP)
                if(curl_handle == NULL) {
                      report("Errore di inizializzazione libcurl!!\n");

                } else {    // Inizializzazione eseguita
                      curl_easy_setopt(curl_handle, CURLOPT_USERNAME,
USER);            // Imposta il nome utente
                      curl_easy_setopt(curl_handle, CURLOPT_PASSWORD,
PASS);            // Imposta la password
                      curl_easy_setopt(curl_handle, CURLOPT_UPLOAD,
1L);        // Abilita upload FTP
                      curl_easy_setopt(curl_handle,
CURLOPT_TRANSFERTEXT, 0);     // Usa la modalità di trasferimento
"binary" (default)
```

```
                        for (i=0; i<files_num; i++) { // Seleziona un file
alla volta
                              file_handle = fopen(eps[i]->d_name, "rb");
      // Apre il file in sola lettura
                              sprintf(remote_file_url, "%s/%s",
REMOTE_TXT_DIR, eps[i]->d_name);    // Crea l'URL completo del file da
spedire
                              curl_easy_setopt(curl_handle, CURLOPT_URL,
remote_file_url);       // Imposta l'URL del file da spedire
                              curl_easy_setopt(curl_handle,
CURLOPT_READDATA, file_handle);     // Imposta file locale da spedire

                              for (k=1; k<4; k++) {// Fa 3 tentativi di
trasmissione per file in caso di fallimento
                                    result =
curl_easy_perform(curl_handle);     // Trasmette il file tramite FTP
                                    if(result == CURLE_OK) {// File
trasmesso con successo
                                          fclose(file_handle);// Chiude
il file
                                          removed = remove(eps[i]-
>d_name);         // Cancella il file in locale
                                          if(removed==0) {// File rimosso
                                                // report("%s Deleted\n",
eps[i]->d_name);
                                          } else {    // File non rimosso
                                                report("%s NOT Deleted\n",
eps[i]->d_name);
                                          }
                                          k = 5;      // Smette di
trasmettere lo stesso file
                                    } else {
                                          if(k>=3) {  // Dopo 3 tentativi
di trasmissione falliti lo segnala
                                                fclose(file_handle);
      // Chiude il file
                                                report("%s ", eps[i]-
>d_name);         // Fa il report col nome del file
                                                report("Transfer failed:
%s\n", curl_easy_strerror(result));
                                          }
                                    }
                              }// Chiude il for per la ritrasmissione
                        }     // Chiude il for per la selezione dei files
                        // report("Trasferimento 'txt' completato!\n");
                  }     // Chiude if curl!= NULL
                  curl_easy_cleanup(curl_handle);     // Chiude la
connessione col server
            } else if (files_num == 0) {  // La cartella è vuota
                  report("Directory %s vuota.\n", LOCALE_TXT_DIR);
            } else {    // Ci sono stati problemi o errori
                  report("Impossibile aprire la directory %s.\n",
LOCALE_TXT_DIR);
            }
            sleep(300);       // Aspetta 5 minuti
      }     // fine while(1)
      return 0;
}     // fine main


/** Funzione che seleziona i files "utili" nella cartella (1=file utile,
0=file da scartare) **/
```

```
int sel_files(const struct dirent *passed_file) {
      // Controlla che il file sia un "bin" o un "txt"
      if ((strstr(passed_file->d_name, ".bin")!=NULL) ||
(strstr(passed_file->d_name, ".txt")!=NULL)) {
            return 1;   // Il file verrà aggiunto alla lista
      } else {          // Non è ne un "bin" ne un "txt"
            return 0;   // Il file verrà scartato
      }
}


/** Funzione che segnala eventuali anomalie o il normale funzionamento
del programma (sullo schermo ed in un file) **/
char report(const char *message, ...) {
      /* Dichiarazione variabili */
      va_list args;           // Lista degli argomenti (variabili)
passati in ingresso alla funzione
      char err_ret = -1;      // Variabile di uscita
      int string_elem;        // Contiene il numero di elementi che
compongono la stringa del messaggio
      char message_string[80] = {0};// Contiene il messaggio sottoforma
di stringa
      FILE *report_file;            // Handler del file report
      va_start(args, message);      // Inizio lista argomenti
      string_elem = vsprintf(message_string, message, args);// Scrive il
messaggio nella stringa "message_string"
      va_end(args);                 // Fine argomenti
      printf("%s", message_string); // Visualizza il messaggio (anche)
sullo schermo
      /* Crea e riempie un file txt con il Report */
      report_file = fopen(REPORT_FILE_NAME, "a");    // Crea file di
report in "append mode"
      fputs_unlocked(message_string, report_file);   // Scrive sul file
(senza bloccarlo)
      fclose(report_file);    // Chiude lo stream verso il file di report
      return (err_ret);
}


/* Funzione che carica il nome del dispositivo */
void device_name_loader(void) {
      /* Dichiarazioni ed inizializzazioni variabili */
      FILE *config_file;            // Handler del file config
      int i = 0;
      /* Apre il file config e carica il nome del dispositivo */
      config_file = fopen(CONFIG_NAME, "r");         // Apre il file di
configurazione "config.txt"
      i = fscanf(config_file, "%s\n", DEVICE_NAME);  // Carica
nell'eseguibile il nome del dispositivo (prima riga)
      fclose(config_file);    // Chiude lo stream verso il file config
}
```

A6. Script "*DB.php*"

Questo è lo script nascosto sul server che viene richiamato a cadenze prefissate per leggere il contenuto dei vari file di testo provenienti dai vari siti (Discusso al Paragrafo 3.2.2.2 ed in Figura 3.19). Trasferisce l'informazione contenuta nei file txt all'interno del database.

```
<?php
$sito = $argv[1];
$corrente=$argv[2];

// Si collega al database. indirizzo, username password.
$connessione = mysql_connect("localhost", "user", "password") or
die(mysql_error());
// Seleziona il database da interpellare.
mysql_select_db("telenergiaDB", $connessione) or die(mysql_error());

$cartella="/data/".strtoupper($sito)."/txt";    // Forma l'URL della
cartella txt

$d = opendir($cartella) or die($php_errormsg); // Apre la cartella txt

while (false !== ($f = readdir($d))) {    // Esamina un file alla volta
nella cartella

      $percorso=$cartella.'/'.$f;
      if (is_file($percorso)) {
            // Forma il TimeStamp dal nome del file
            list($anno,$mese,$giorno,$ora,$minuto,$est) = split('[_.-]',
$f);
            $timestamp=$anno."-".$mese."-".$giorno."
".$ora.":".$minuto.":00";
            // Apre il file e carica la prima riga
            $file = fopen($percorso, "r") or exit("Impossibile aprire il
file!");
            list($frequenza,$Vr,$Vs,$Vt,$Vn,$THD)=split(';',
fgets($file));

      $str_query1="time_stamp,frequenza,VeffR,VeffS,VefffT,VeffN,THD";
            // Qui  vengono scritti i valori ricavati dal file

      $str_query2="'".$timestamp."','".$frequenza."','".$Vr."','".$Vs."',
'".$Vt."','".$Vn."','".$THD."'";

            // Carica tutte le informazioni del file e prepara le
stringhe per la query al DB
            for ($i = 1; $i <= 24; $i++) {
                  if ($corrente==0){

      list($armRmod,$armRfas,$armSmod,$armSfas,$armTmod,$armTfas)=split('
;',fgets($file));

      $str_query1=$str_query1.",Rarm".$i."mod,Rarm".$i."fas,Sarm".$i."mod
,Sarm".$i."fas,Tarm".$i."mod,Tarm".$i."fas";

      $str_query2=$str_query2.",".$armRmod.",".$armRfas.",".$armSmod.",".
```

```
$armSfas.",".$armTmod.",".$armTfas;
                  }elseif($corrente==1){

      list($armRmod,$armRfas,$armSmod,$armSfas,$armTmod,$armTfas,$armRImo
d,$armRIfas,$armSImod,$armSIfas,$armTImod,$armTIfas)=split(';',fgets($fil
e));

      $str_query1=$str_query1.",Rarm".$i."mod,Rarm".$i."fas,Sarm".$i."mod
,Sarm".$i."fas,Tarm".$i."mod,Tarm".$i."fas,RIarm".$i."mod,RIarm".$i."fas,
SIarm".$i."mod,SIarm".$i."fas,TIarm".$i."mod,TIarm".$i."fas";

      $str_query2=$str_query2.",".$armRmod.",".$armRfas.",".$armSmod.",".
$armSfas.",".$armTmod.",".$armTfas.",".$armRImod.",".$armRIfas.",".$armSI
mod.",".$armSIfas.",".$armTImod.",".$armTIfas;
                  }
            }

            try{ // Finisce la stringa per l'inserimento dei dati nel DB
                  $stato=str_replace(array("\n","\r"), "",fgets($file));
                  if ($stato == "corrotto"){
                        $str_query1=$str_query1.",controllo";
                        $str_query2=$str_query2.",1";
                  }elseif($stato == "OK"){
                        echo $stato;
                        $str_query1=$str_query1.",controllo";
                        $str_query2=$str_query2.",0";
                  }
            }
            catch (Exception $e) {};

            fclose($file); // Chiude il file

            $query="INSERT INTO ".$sito."(".$str_query1.")
VALUES(".$str_query2.")";
        // Esegue la query di INSERT nel DB
            mysql_query($query) or die(mysql_error());
            // Cancella il file se la scrittura nel DB è andata a buon
fine
            unlink($percorso);
      }
}
mysql_close($connessione);    // Chiude la connessione al DB
closedir($d);           // Chiude la directory

?>
```

A7. Script Python

Questo script, lanciato a cadenze regolari sul BeagleBone Black, permette di estrarre i valori di CPU, RAM e flash attualmente utilizzati sul uPC e li trasmette al server che li registra nel DB, insieme all'indirizzo IP della chiamata (Paragrafo 3.3.2 e Figura 3.29).

```
# Script per estrarre i valori di CPU, RAM e flash attualmente utilizzati
# Poi inoltra i valori al server che registra anche l'indirizzo ip
# Quando viene chiamato da shell bisogna fornirgli il nome del sito come
parametro
# Chiamare con il comando "python nome_script.py nome_sito"

#!/usr/bin/python

# Importa le librerie
import subprocess
import sys

# Verifica che sia stato inserito il nome del sito come parametro
if len(sys.argv) != 2:
      print "I parametri non sono corretti!"
      print "Riprova e inserisci il nome del sito."
      sys.exit(0)

# Lancio il comando "cat /proc/loadavg"
p = subprocess.Popen(["cat", "/proc/loadavg"], stdout=subprocess.PIPE)
(output, err) = p.communicate()
# Suddivido l'uscita nei vari termini
output = output.split( )
# Valore della CPU utilizzata (in percentuale, mediata nei 15 minuti)
cpu_usata = output[2]
#print cpu_usata

output = 0  # Per sicurezza resetto la variabile

# Lancio il comando "free"
p = subprocess.Popen("free", stdout=subprocess.PIPE)
(output, err) = p.communicate()
# Suddivido l'uscita nelle varie righe
output = output.split("\n")
# Suddivido la riga dei valori della Ram nei vari termini
output = output[1].split( )
# Calcolo la percentuale di RAM utilizzata
ram_usata=float(output[2])/float(output[1])*100
# Converto il float in una stringa e lo arrotondo all'intero piu' vicino
ram_usata = "%.0f" % ram_usata
# Valore della RAM utilizzata
#print ram_usata

output = 0  # Per sicurezza resetto la variabile

# Lancio il comando "df -h"
p = subprocess.Popen(["df", "-h"], stdout=subprocess.PIPE)
(output, err) = p.communicate()
# Suddivido l'uscita nelle varie righe
output = output.split("\n")
# Suddivido la riga dei valori della flash nei vari termini
```

```
output = output[1].split( )
# Valore della flash utilizzata (in percentuale, senza %)
flash_usata = output[4].replace("%", "")
#print flash_usata

# Preparazione del link
link="http://ip.server/ip_logger.php?sito="+sys.argv[1]+"&cpu="+cpu_usata
+"&ram="+ram_usata+"&flash="+flash_usata
#print link
# Connessione e comunicazione dei parametri al server
subprocess.call(["curl", "--silent", link])
```

A8. Script “Ip_logger.php”

Questo è lo script che registra sul DB gli indirizzi IP dei vari siti e gli altri parametri che gli vengono passati quando viene invocato.

```
<?php // Logger degli ip dei probe
$sito = $_GET['sito'];
$cpu = $_GET['cpu'];
$ram = $_GET['ram'];
$flash = $_GET['flash'];
$data = date("Y-m-d");
$ora = date("G:i:s");
$ip_sito = $_SERVER['REMOTE_ADDR'];

switch($sito) {
      case NULL:
            echo "NULL\n\r";
            break;

      // Siti permessi:
      case (($sito=="sito1") || ($sito=="sito2") || ($sito=="sito3") ||
($sito=="sito4") || ($sito=="sito5") || ($sito=="sito6") ||
($sito=="sito7") || ($sito=="sito8")):

            // Mi collego al database. indirizzo, username e password.
            mysql_connect("localhost", "user", "password") or
die(mysql_error());
            // seleziono il database da interpellare.
            mysql_select_db("probe_info") or die(mysql_error());

            if((isset($_GET['cpu'])) && (isset($_GET['ram'])) &&
(isset($_GET['flash']))) {
                  $query="UPDATE ip_address SET ip='".$ip_sito."',
data='".$data." ".$ora."', cpu='".$cpu."', ram='".$ram."',
flash='".$flash."' WHERE sito='".$sito."'";
            } else {
                  $query="UPDATE ip_address SET ip='".$ip_sito."',
data='".$data." ".$ora."' WHERE sito='".$sito."'";
            }
            // eseguo la query di UPDATE nel DB
            mysql_query($query) or die(mysql_error());
            mysql_close();          // chiudo il DB

            echo "OK\n\r";
            break;

      default:
            echo "NONE\n\r";
            break;
}
?>
```

A9. Interfaccia "*ip_table*"

(Mostrata al Paragrafo 3.3.3 ed in Figura 3.30).

```
<!DOCTYPE html>
<html>
<head><title> Ip Table v2</title></head>
<body>
<br />

<?php
$controllo = $_GET['param'];        // Parametro di controllo in ingresso

// Mi collego al database. indirizzo, username e password.
mysql_connect("localhost", "user", "password") or die(mysql_error());
// seleziono il database da interpellare.
mysql_select_db("probe_info") or die(mysql_error());

// Prepara la query
//$query = "SELECT sito, comune, data FROM ip_address ORDER BY comune,
sito";
$query = "SELECT sito, comune, data FROM ip_address ORDER BY ID";

/* Prepara la tabella */
echo '<table border=1 align=center cellspacing=3 cellpadding=8>';
echo '<tr align=center><td><b><u> Date </b></u></td><td><b><u> Time
</b></u></td><td><b><u> Site </b></u></td>';
if($controllo=="yes") {
      echo '<td><b><u> IP </b></u></td>';
      //$query = "SELECT sito, comune, ip, data FROM ip_address ORDER BY
comune, sito";
      $query = "SELECT sito, comune, ip, data FROM ip_address ORDER BY
ID";
}
echo '<td><b><u> Connection </b></u></td><td><b><u> Last conn.
</b></u></td><td><b><u> Last packet </b></u></td>';
echo '<td><b><u> Elapsed time </b></u></td><td><b><u> Status
</b></u></td><td><b><u> Info </b></u></td></tr>';

// Interroga il DB
$result = mysql_query($query) or die(mysql_error()); // eseguo la query
di SELECT nel DB

// Cambio il database da interpellare per le prossime query sui last
packets
mysql_select_db("telenergiaDB") or die(mysql_error());

while($row = mysql_fetch_array($result)){

      $problemi = 0;    // Reset conteggio problemi

      // Cambia il formato della data
      $row["data"] = date('d-m-Y H:i:s', strtotime($row["data"]));

      // Separa la data dall'ora
      $a = explode(' ', $row["data"]);
      $data = $a[0];
      $ora = $a[1];

      echo '<tr align=center><td>'.$data.'</td>';
```

```
	echo '<td>'.$ora.'</td>';
	echo '<td>'.$row["comune"].' - '.$row["sito"].'</td>';
	if($controllo=="yes") {
		echo '<td>'.$row["ip"].'</td>';		// Visualizza gli ip
	}

	/* Controlla se i siti non rispondono da più di 30 minuti: */
	// Calcola il tempo trascorso (in secondi) dall'ultima connessione

	$elapsed_time = time() - strtotime($row["data"]);

	if($elapsed_time < 1900) {
		echo '<td><font color=green> On-line </font></td>';
		echo '<td><font color=green>'.round($elapsed_time/60).'
min</font></td>';
	} else {
		echo '<td><font color=red> Check!! </font></td>';
		echo '<td><font color=red>'.round($elapsed_time/60).'
min</font></td>';
		$problemi++;
	}

	// Prepara la query per il last_packet
	$query = "SELECT time_stamp FROM ".strtolower($row["sito"])." ORDER
BY ID DESC LIMIT 1";
	// eseguo la query di SELECT nel DB
	$last_packet = mysql_query($query);

	// Query eseguita con successo (l'ultimo pacchetto esiste ed è
stata estratta l'ora)
	if($last_packet != FALSE) {

		$last_packet = mysql_fetch_array($last_packet);
		$last_packet = $last_packet[0];		// Prende solo il
time_stamp

		// Calcola il tempo trascorso (in secondi) dall'ultimo
pacchetto spedito-elaborato
		$elapsed_time = NULL;
		$elapsed_time = strtotime(gmdate('Y-m-d H:i')) -
strtotime($last_packet);

		// Cambia il formato della data
		$last_packet = date('d-m-Y H:i:s', strtotime($last_packet));

		if($elapsed_time < 720) {	// Sono passati più di 12
minuti?
			echo '<td><font color=green>'.$last_packet.'
UTC</font></td>';
			echo '<td><font color=green>'.round($elapsed_time/60).'
min</font></td>';
		} else {
			echo '<td><font color=red>'.$last_packet.'
UTC</font></td>';
			echo '<td><font color=red>'.round($elapsed_time/60).'
min</font></td>';
			$problemi++;
		}
	}

	// L'ultima query ha avuto esito negativo (Non è disponibile l'ora
dell'ultimo pacchetto)
```

```
      else {
            echo '<td><font color=red> - - - - </font></td>';
            echo '<td><font color=red> - - - - </font></td>';
            $problemi++;
      }

      if($problemi == 0) {          // Sul sito ci sono problemi?
            echo '<td><font color=green> OK </font></td>';
      } else {
            echo '<td><font color=red> Check!! </font></td>';
      }
      $info="window.open('code_panel.php?site=".$row["sito"]."')";
      echo '<td><input type="button" name="ProbeInfo" value="Info"
onclick="'.$info.'"/></td>';

      echo '</tr>';     // Fine riga
}
echo "</table>";  // Fine tabella
mysql_close();    // Chiudo il DB

?>

<div align=center><table border=0 align=center cellspacing=3
cellpadding=4><tr>
<td align=left>
<?php
echo "<font color=blue><u> Server Local Time:</u> ";
echo date('d-m-Y H:i:s')."         
 </font>";
?></td>

<td align=center><input type="button" name="update" value=" Update "
onclick="document.location.reload()"/></td>
<td align=right>
<?php
echo "<font color=blue>           
<u>Server UTC Time:</u> ";
echo gmdate('d-m-Y H:i:s')." UTC</font>";

if($controllo=="yes") { // Visualizza anche il pulsante "Probe State"
      echo '</td></tr><tr><td align=right>';
      $link = "'panel1.html'";
      echo '<input type="button" name="ToPanel" value="Link to Panel"
onclick="window.open('.$link.')"/>
       </td><td align=center>';
      $link = "'code_panel.php'";
      echo '<input type="button" name="ProbeInfo" value="Probe Info"
onclick="window.open('.$link.')"/>
      </td><td align=left> ';
      $link = "'pro_stat.html'";
      echo '<input type="button" name="ProState" value="Probe State"
onclick="window.open('.$link.')"/>';
} else {          // No pulsante "Probe State"
      echo '</td></tr><tr><td></td><td align=center>';
      $link = "'panel1.html'";
      echo '<input type="button" name="ToPanel" value="Link to Panel"
onclick="window.open('.$link.')"/>
        ';
      $link = "'code_panel.php'";
      echo '<input type="button" name="ProbeInfo" value="Probe Info"
onclick="window.open('.$link.')"/>';
}
```

```
?>

</td></tr></table>
</div>

</body>
</html>
```

A10. Interfaccia “*code_panel*”

(Illustrata al Paragrafo 3.3.5 ed in Figura 3.32).

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html;charset=ISO-8859-1">
<title> Code Panel </title>
      <link rel="stylesheet" href="./lightbox/colorbox.css" />
      <script src="./lightbox/jquery.min.js"></script>
      <script src="./lightbox/jquery.colorbox-min.js"></script>
</head>
<body>
<div align=center><form method='post' action="code_panel.php">
<table border=1 align=center cellspacing=2 cellpadding=4>
<td><label><b>Sito:</b>
<select name="formSite" id="formSite" size="1">
      <option value="ALL" selected="selected">ALL</option>
      <option value="ALL_ROMA">ALL ROMA</option>
      <option value="CORVIALE">ROMA-Corviale</option>
      <option value="INVIOLATELLA">ROMA-Inviolatella</option>
      <option value="PIETRALATA">ROMA-Pietralata</option>
      <option value="TORPAGNOTTA">ROMA-Tor Pagnotta</option>
      <option value="LAB">ROMA-Laboratorio</option>
      <option value="BEAGLE1">Beagle1-Prove</option>
</select> </label></td>

<td><label><b>Data inizio:</b>
<select name="startDate" id="startDate" size="1">
      <option value="01" selected="selected">Gennaio</option>
      <option value="02">Febbraio</option>
      <option value="03">Marzo</option>
      <option value="04">Aprile</option>
      <option value="05">Maggio</option>
      <option value="06">Giugno</option>
      <option value="07">Luglio</option>
      <option value="08">Agosto</option>
      <option value="09">Settembre</option>
      <option value="10">Ottobre</option>
      <option value="11">Novembre</option>
      <option value="12">Dicembre</option>
</select> </label>
<select name="startYear" id="startYear" size="1">
      <option value="2014" selected="selected">2014</option>
      <option value="2015">2015</option>
      <option value="2016">2016</option>
</select>
</td>

<td><label><b>Data Fine:</b>
<select name="stopDate" id="stopDate" size="1">
      <option value="01">Gennaio</option>
      <option value="02">Febbraio</option>
      <option value="03">Marzo</option>
      <option value="04">Aprile</option>
      <option value="05">Maggio</option>
      <option value="06">Giugno</option>
      <option value="07">Luglio</option>
      <option value="08">Agosto</option>
```

```
      <option value="09">Settembre</option>
      <option value="10">Ottobre</option>
      <option value="11">Novembre</option>
      <option value="12" selected="selected">Dicembre</option>
</select> </label>
<select name="stopYear" id="stopYear" size="1">
      <option value="2014">2014</option>
      <option value="2015">2015</option>
      <option value="2016" selected="selected">2016</option>
</select>
</td>

<td>
<input type='submit' value=" Cerca "/>
</td></table></form></div>
<br />
<div>

<?php
// Funzione per la visualizzazione delle singole tabelle dei siti
function visualizza($site,$staYear,$staDate,$stoYear,$stoDate) {

      // Prepara la query
      $query = "SELECT * FROM ".$site."_CODE WHERE data BETWEEN
'".$staYear."-".$staDate."-01' AND '".$stoYear."-".$stoDate."-31' ORDER
BY data DESC";

      // Interroga il DB
      $result = mysql_query($query) or die(mysql_error()); // eseguo la
query di SELECT nel DB

      $a = mysql_num_rows($result); // Numero delle righe restituite

      if($a > 0) {
            $gsite = "'".$site."'";
            echo '<br/><b><font color=blue><a
href="code_panel.php?site='.$site.'"> '.$site.'</a></font></b> (righe:
'.$a.')     <input type="button" value="INSERT"
onclick="crea_riga('.$gsite.')"/><br/>'."\n";
            echo '<table border=1 cellspacing=3 cellpadding=8>'."\n";
            echo '<tr><td width="136" align=center> Date & time (local)
</td><td align=center> Code </td><td align=center> Operator </td><td
align=center> Note </td><td align=center> Modify </td><td align=center>
Delete </td></tr>'."\n";

            while($row = mysql_fetch_array($result)){

                  $gdata = $row["data"];
                  $b_param1 =
"'".$site."','".$gdata."','".$row["code"]."','".$row["operator"]."','".$r
ow["note"]."','update'";
                  $b_param2 =
"'".$site."','".$gdata."','".$row["code"]."','".$row["operator"]."','".$r
ow["note"]."','delete'";
                  echo
'<tr><td>'.$row["data"].'</td><td>'.$row["code"].'</td><td>'.$row["operat
or"].'</td><td>'.$row["note"].'</td>';
                  echo '<td><input type="button" value="MOD"
onclick="aggiorna_riga('.$b_param1.')"/></td>';
                  echo '<td><input type="button" value="DEL"
onclick="aggiorna_riga('.$b_param2.')"/></td>';
                  echo '</tr>'."\n";
```

```
            }

            echo "</table>"."\n";
      }

return 1;
}
?>

<?php
$formSite = $_POST['formSite'];           // Parametri in ingresso
$startDate = $_POST['startDate'];
$startYear = $_POST['startYear'];
$stopDate = $_POST['stopDate'];
$stopYear = $_POST['stopYear'];

// Mi collego al database. indirizzo, username e password.
mysql_connect("localhost", "user", "password") or die(mysql_error());
// seleziono il database da interpellare.
mysql_select_db("probe_info") or die(mysql_error());

if(isset($_GET['site'])) {          // Parametro proveniente con il
metodo Get
      echo "Sito selezionato: ".$_GET['site'].". Periodo d'interesse: da
01-01-2014 a 31-12-2016.<br />";
      //$get_site = $_GET['site'];
      visualizza($_GET['site'],"2014","01","2016","12");

} elseif(!empty($_POST)) {                    // I parametri provengono
con il metodo Post
      echo "Sito selezionato: ".$formSite.". Periodo d'interesse: da 01-
".$startDate."-".$startYear." a 31-".$stopDate."-".$stopYear.".<br />";

      if($formSite=="ALL"){   // Visualizza tutti i record

      visualizza("CORVIALE",$startYear,$startDate,$stopYear,$stopDate);

      visualizza("INVIOLATELLA",$startYear,$startDate,$stopYear,$stopDate
);

      visualizza("PIETRALATA",$startYear,$startDate,$stopYear,$stopDate);

      visualizza("TORPAGNOTTA",$startYear,$startDate,$stopYear,$stopDate)
;
            visualizza("LAB",$startYear,$startDate,$stopYear,$stopDate);

      visualizza("BEAGLE1",$startYear,$startDate,$stopYear,$stopDate);
      } elseif($formSite=="ALL_ROMA"){    // Visualizza tutti i record di
Roma

      visualizza("CORVIALE",$startYear,$startDate,$stopYear,$stopDate);

      visualizza("INVIOLATELLA",$startYear,$startDate,$stopYear,$stopDate
);

      visualizza("PIETRALATA",$startYear,$startDate,$stopYear,$stopDate);

      visualizza("TORPAGNOTTA",$startYear,$startDate,$stopYear,$stopDate)
;
            visualizza("LAB",$startYear,$startDate,$stopYear,$stopDate);
      } else {
            // Visualizza il sito specificato
```

```
      visualizza($formSite,$startYear,$startDate,$stopYear,$stopDate);
      }
} else {
      echo "Selezionare sito e periodo d'interesse.<br />";

}

mysql_close();          // chiudo il DB
?>

</div>
<script>
      // Funzione che crea il form per Aggiungere una riga nel DB
      function crea_riga(sito) {
            var data = new Date();
            // Creo una stringa nel formato "2014-10-11 21:09:24"
            var data_string = data.getFullYear()+"-
"+(data.getMonth()+1)+"-"+data.getDate()+"
"+data.getHours()+":"+data.getMinutes()+":"+data.getSeconds();
            aggiorna_riga(sito,data_string,'Code','User','Inserire
note','new');
      }

      // Funzione che crea il form per modificare i campi del DB su una
colorbox
      function aggiorna_riga(sito,timestamp,code,oper,note,entry) {
            var param = '"'+sito+'","'+entry+'"';
            var tabella = "<b>Modifica di: <font
color=blue>"+sito+"</font></b>.  <span
id='risp_server2'></span><br/><br/><form method='post'
name='info_probe'><table><tr><td><b>Date & time
(local):</b></td><td><b>Code:</b></td><td><b>Operator:</b></td></tr><tr><
td><input type='text' name='timestamp'
value='"+timestamp+"'/></td><td><input type='text' name='code'
value='"+code+"'/></td><td><input type='text' name='operator'
value='"+oper+"'/></td></tr></table><br/><b>Note:</b><br/><textarea
name='note' rows='3' cols='80'>"+note+"</textarea><br/><input
type='button' value='Update DB' onclick='login("+param+")'/></form>";
            $.colorbox({html:tabella});
            if(entry=="new"){
                  document.getElementById("risp_server2").innerHTML =
"<b> Comando: <font color=green>Nuova riga</font></b>";
            } else if(entry=="update"){
                  document.getElementById("risp_server2").innerHTML =
"<b> Comando: <font color=green>Aggiorna riga</font></b>";
            } else if(entry=="delete"){
                  document.getElementById("risp_server2").innerHTML =
"<b> Comando: <font color=red>Cancella riga</font></b>";
            }
      }

      // Funzione che crea il form per il login prima di aggiornare il DB
      function login(site,entry) {
            var tm = document.info_probe.timestamp.value;
            var cod = document.info_probe.code.value;
            var op = document.info_probe.operator.value;
            var not = document.info_probe.note.value;
            var parametri =
'"'+site+'","'+tm+'","'+cod+'","'+op+'","'+not+'","'+entry+'"';
            var form_login = "<form method='post'
name='formLogin'><table><tr><td><b>User: </b></td><td><input type='text'
```

```
name='userid'/></td></tr><tr><td><b>Pssw: </b></td><td><input
type='password' name='psw'/></td></tr><tr><td></td></tr><tr><td><input
type='button' value=' OK ' onclick='log("+parametri+")'/></td><td><span
id='risp_server1'></span></td></tr></table></form><br/>";
            $.colorbox({html:form_login});
      }

      // Carica tutti i parametri e li manda al php (ajax) per modificare
il DB
      function log(site,tm,cod,op,not,entry) {
            var usr = document.formLogin.userid.value;
            var pas = document.formLogin.psw.value;

            if (window.XMLHttpRequest){          // Codice per IE7+,
Firefox, Chrome, Opera, Safari
                  xmlhttp=new XMLHttpRequest();
            }
            else{       // Codice per IE6, IE5
                  xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
            }

            xmlhttp.onreadystatechange=function(){
                  if (xmlhttp.readyState==4 && xmlhttp.status==200) {
                        var risposta = xmlhttp.responseText;
                        if (risposta=="updated") {
                              // Segnalo il corretto aggiornamento (o
creazione) della riga del DB
                              var up_message = "La tabella di <b><font
color=blue>"+site+"</font></b> &egrave; stata aggiornata con i seguenti
valori:<br/><br/><table><tr><td><b>Date & time
(local):</b></td><td></td><td><b>Code:</b></td><td></td><td><b>Operator:<
/b></td></tr><tr><td>"+tm+"</td><td></td><td>"+cod+"</td><td></td><td>"+o
p+"</td></tr></table><br/><b>Note:</b><br/>"+not+"<br/>";
                              $.colorbox({html:up_message,
onClosed:function(){location.reload(true)}});

                        } else if (risposta=="log_error") {
                              // Segnalo che c'è stato un errore sul
login e rimango sul form del login

      document.getElementById("risp_server1").innerHTML = "<b><font
color=red> Credenziali non valide!!</b></font>";

                        } else if (risposta=="data_error") {
                              // Segnalo un'anomalia nell'aggiornamento
del DB
                              aggiorna_riga(site,tm,cod,op,not,entry);

      document.getElementById("risp_server2").innerHTML = "<b><font
color=red>Errore nell'aggiornamento del DB!!</b></font>";

                        } else if (risposta=="deleted") {
                              // Segnalo che la tabella è stata rimossa
correttamente
                              var del_message = "La tabella di <b><font
color=blue>"+site+"</font></b> con i seguenti
valori:<br/><br/><table><tr><td><b>Date & time
(local):</b></td><td></td><td><b>Code:</b></td><td></td><td><b>Operator:<
/b></td></tr><tr><td>"+tm+"</td><td></td><td>"+cod+"</td><td></td><td>"+o
p+"</td></tr></table><br/><b>Note:</b><br/>"+not+"<br/><div
align=center><br/><b><u><font color=red>&egrave; stata
RIMOSSA!!</font></u></b></div>";
```

```
                                $.colorbox({html:del_message,
onClosed:function(){location.reload(true)}});
                            }
                        }
                    }
                    xmlhttp.open('post','mod.php',true);
                    // Imposto gli header per il metodo post
                    xmlhttp.setRequestHeader('content-type', 'application/x-www-
form-urlencoded');

            xmlhttp.send('site='+site+'&data='+tm+'&code='+cod+'&oper='+op+'&no
te='+not+'&userid='+usr+'&psw='+pas+'&entry='+entry);
            }
</script>
</body>
</html>
```