Agenzia nazionale per le nuove tecnologie,
l'energia e lo sviluppo economico sostenibile

MINISTERO DELLO SVILUPPO ECONOMICO

Ricerca di Sistema elettrico

Analisi e sviluppo di un prototipo di rete di misuratori per la gestione multi-utility con funzionalità avanzate per la gestione dell'energia

C. Landi, D. Gallo, M. Luiso, F. Clarizia, R. Rinaldi

Seconda Università degli Studi di Napoli
SUN
DIII SUN
Dipartimento di Ingegneria
Industriale e dell'Informazione

RdS/PAR2014/022

ANALISI E SVILUPPO DI UN PROTOTIPO DI RETE DI MISURATORI PER LA GESTIONE MULTI-UTILITY CON FUNZIONALITÀ AVANZATE PER LA GESTIONE DELL'ENERGIA

C. Landi, D. Gallo, M. Luiso, F. Clarizia, R. Rinaldi (Seconda Università di Napoli , Dipartimento di Ingegneria Industriale e dell'Informazione)

Settembre 2015

Report Ricerca di Sistema Elettrico
Accordo di Programma Ministero dello Sviluppo Economico – ENEA
Piano Annuale di Realizzazione 2014
Area: Razionalizzazione e risparmio nell'uso dell'energia
Progetto C1: Risparmio di energia elettrica nei settori: civile, industria e servizi
Responsabile del Progetto: ing. Ilaria Bertini, ENEA

Il presente documento descrive le attività di ricerca svolte all'interno dell'Accordo di collaborazione *"Sperimentazione e validazione in campo di smart meter e confronto prestazionale con misuratori di tipo convenzionale"*

Responsabile scientifico ENEA: ing. Giorgio Graditi

Responsabile scientifico SUN: prof. ing. Carmine Landi

Indice

Introduzione

La gestione delle misurazioni dei flussi di energia elettrica e dei parametri ad essi connessi risulta oggi di fondamentale importanza non più unicamente ai fini della tariffazione, ma anche per il coordinamento e la gestione integrata dell'energia con l'obiettivo di ottenere una razionalizzazione del suo uso ed una conseguente riduzione dei consumi, sia nelle attività industriali sia in quelle commerciali o domestiche. La conoscenza in tempo reale dei profili di consumo consente a chi gestisce le reti energetiche di realizzare meccanismi di maggiore dinamicità, flessibilità, decentralizzazione e interattività nella organizzazione delle reti stesse ed, inoltre, permette a chi utilizza l'energia una maggiore consapevolezza di quanto si stia consumando.

I temi sviluppati nell'ambito dell'accordo di collaborazione tra ENEA e Seconda Università degli studi di Napoli (SUN) riguardano lo sviluppo di metodologie di misura avanzate per reti di distribuzione asservite all'alimentazione di distretti energetici e la definizione delle specifiche dell'architettura di "concentratori dati intelligenti" che consentono l'interazione tra generazione e utilizzazione (prevedendo altresì accesso da remoto e la relativa pubblicazione dati).

Tale attività si è estrinsecata attraverso la progettazione di un prototipo di rete di misuratori per la gestione multi utility con funzionalità avanzate per la gestione dell'energia e la realizzazione e caratterizzazione del prototipo di rete di misuratori. L'attività è stata effettuata secondo diverse fasi che possono essere ricondotte a: i) studio ed individuazione di architetture e protocolli di comunicazione standard atti a consentire la intercambiabilità dei misuratori attraverso l'adozione di un front-end e di un meta-linguaggio di comunicazione; ii) messa a punto di procedure di configurazione, calibrazione e diagnostica dei diversi misuratori; iii) l'adozione di un uno strato software (layer) che fungesse da clearing-house per la generazione di un meta-dato, utilizzabile da un nodo di livello superiore; iv) individuazione di "indici di prestazione" quali: accuratezza della misura, resilienza ad errori di comunicazione, tempo di latenza sulla comunicazione, sicurezza del dato, minimo intervallo di lettura, per la qualificazione di reti di misuratori; v) verifica delle prestazioni della rete sviluppata con caratterizzazione metrologica attraverso sperimentazione in laboratorio e sul campo con confronto con dispositivi di misura di tipo commerciale.

L'attività svolta nel presente contratto si inserisce a valle di precedenti fasi di studio ed implementazione dei diversi (singoli) dispositivi di misura effettuata nell'ambito dei precedenti contratti. Più in particolare, in precedenza era stato sviluppato atopicamente un misuratore che non prevedeva la possibilità di integrare ed interagire con le diverse configurazioni topologiche necessarie (mancanza di una struttura gerarchica di controllo con diversi punti di sensing). Nel

corso del presente contratto si è, invece, proceduto alla progettazione e realizzazione dell'intero sistema per il monitoraggio contemporaneo dei diversi hot points (che possono essere sia i carichi che i generatori) anche con l'individuazione e lo sviluppo di protocolli propri delle reti di calcolatori e la messa a disposizione di tali informazioni (attraverso un Database remoto) agli utenti a vario titolo interessati alle informazioni raccolte. Inoltre, un'ulteriore azione ha riguardato la messa in sicurezza dell'intera catena di scambio di informazioni tra i diversi dispositivi presenti (studio ed implementazione di protocolli di scambio dati sicuri su ogni dispositivo e per ogni comunicazione). Infine, è stato effettuato un confronto prestazionale tra il dispositivo/sistema realizzato e alcuni dispositivi commerciali presenti ad oggi sul mercato. A riguardo sono stati monitorati i consumi energetici di tipo elettrico, oggetto di tariffazione di un complesso commerciale ubicato nel comune di Aversa (NA). Più precisamente è stato eseguito in maniera continuativa il monitoraggio dei suddetti consumi registrando ogni minuto gli assorbimenti medi in termini di potenza attiva e reattiva. Tutte le apparecchiature servite sono state impiegate nelle modalità e nei tempi consueti tipici della normale attività lavorativa. Allo scopo del confronto le misurazioni sono state effettuate in parallelo sia da un prototipo dello Smart Meter sviluppato ed implementato sia da un misuratore di tipo commerciale, ottenendo prestazioni soddisfacenti ed il linea con quelle attese.

1 Progettazione di un prototipo di rete di misuratori per la gestione multi-utility con funzionalità avanzate per la gestione dell'energia

Una rete di misuratori deve essere un sistema ampiamente scalabile di acquisizione dati, capace di analizzare diversi aspetti della rete elettrica (e non solo i consumi energetici) ma deve necessariamente elaborare i dati acquisiti in linea con la loro acquisizione (real time). Deve altresì prevedere una semplice ed intuitiva interfaccia utente per la visualizzazione dei dati e poter agire direttamente su di essi, se richiesto, così come implementare diverse periferiche di I/O che ne permettano la connettività per lo scambio di dati. La rete implementata ha una struttura ad albero gerarchico di tipo clusterizzata, prevedendo diversi livelli. In particolare per la rete realizzata ne sono stati previsti tre: i) il misuratore, ii) il concentratore di area ed iii) un concentratore generale. Tale struttura, riportata in Figura 1, risulta espandibile con l'inserimento di più dispositivi allo stesso livello ovvero l'inserimento di più livelli con capacità elaborative man mano sempre più elevate salendo attraverso l'albero. La modularità e la scalabilità con la possibilità di interconnettere, a diversi livelli, i singoli nodi operativi consente, quindi, di essere applicata anche a reti molto estese.

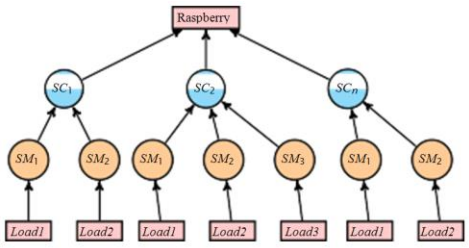


Figura 1: Smart Meters Network

Le funzionalità individuate sono quindi: misuratori di livello 0 (Smart Meter) per effettuare le misure/acquisire dati sulla rete elettrica, nodi aggregatore (Smart Concentrator) di livello 1 in grado di recuperare le misure dei singoli Smart Meter afferenti alla propria sottorete e nodo Interfaccia di livello 2 (Micro Computer/Raspberry) in grado di fornire interfacciamento verso gli utenti finali.

1.1 Studio ed Individuazione di protocolli di comunicazione atti a consentire la intercambiabilità dei misuratori prodotti da differenti costruttori

Considerando i diversi livelli operativi di ogni singolo dispositivo presente nella rete di misuratori (sistema gerarchico), è stato necessario individuare due tipi di protocolli che permettessero lo scambio di dati in modo efficiente ed efficace, ma che potessero allo stesso tempo rendere l'intero sistema aperto (interconnettere dispositivi di costruttori diversi). Obiettivo di questa fase è stato dunque individuare i protocolli - necessariamente standard - che garantissero la trasmissione senza perdite (nello specifico con controllo di flusso e controllo di errore), ma che fossero in grado di non appesantire eccessivamente i dispositivi (che hanno come compito principale l'esecuzione delle misure). Dopo lo studio della letteratura di settore, sono stati scelti il protocollo CAN per le comunicazioni tra livello 0 e livello 1 ed il protocollo HTTP (basato su stack TCP/IP) tra livello 1 e livello 2. Il protocollo HTTP è stato, altresì, utilizzato anche per fornire l'accesso ai dati per l'utente finale (interfaccia web user friendly).

1.1.1 Introduzione ai principali protocolli di comunicazione

Tra le tecnologie più diffuse (intese come protocolli di comunicazione) impiegate in reti di Smart Meter troviamo la RS-232 e il CAN (Controller Area Network) che implementano protocolli di tipo seriale asincrono.

Seriale significa che i bit che costituiscono l'informazione sono trasmessi uno alla volta, in successione, su un solo "filo" di connessione. Questo termine è in genere contrapposto a *Parallelo*: in questo caso i dati sono trasmessi contemporaneamente su più fili (e.g. bus PCI nei computer è un bus parallelo a 32 bit). Da notare che una trasmissione seriale non è a priori più lenta di una parallela: se da un lato su di un filo possono passare meno informazioni che su 32 questo viene bilanciato dalla difficoltà di controllare lo skew (disallineamento temporale tra i vari segnali) dei molti trasmettitori in un bus parallelo. Per esempio in una fibra ottica o in un cavo FireWire (standard seriali) le informazioni transitano ad una velocità paragonabile a quella di un bus PCI parallelo. In ogni caso la scelta nei sistemi di Smart metering (SM) di adottare standard seriali è legata alla necessità di semplificare i cablaggi, minimizzare errori di trasmissione/ricezione dei messaggi, alla richiesta di bit-rate non elevati (max. centinaia di kbits/s).

Il termine asincrono indica che i dati sono trasmessi senza l'aggiunta di un segnale di clock, cioè di un segnale comune per sincronizzare la trasmissione con la ricezione; sia il trasmettitore che il ricevitore sono comunque dotati di un clock locale per poter interpretare i dati. La sincronizzazione dei due clock è necessaria ed è fatta in corrispondenza della prima transizione sulla linea dei dati.

Se la trasmissione dati è bidirezionale ma non avviene contemporaneamente nelle due direzioni si parla di comunicazione *Half-duplex*: un dispositivo (ricevitore Rx) ascolta e l'altro (trasmettitore Tx) emette segnali. Quando è necessario si scambiano i ruoli. *Full-duplex* indica che la trasmissione è bidirezionale e contemporanea. In questo caso sono necessari due fili oppure qualche altro sistema (divisione di frequenza, divisione di codice) per distinguere i due messaggi contemporanei nelle due direzioni. Se la trasmissione è sempre in un solo verso, si parla di *Simplex*. Tipicamente i vari nodi che costituiscono la rete del sistema di controllo fungono sia da trasmettitori che ricevitori - protocollo *Duplex*.

Se nel bus uno solo dei nodi svolge la funzione di gestore della rete (controlla e gestisce l'accesso al bus) si parla di protocollo *Master/Slave* in cui il predetto nodo funge da *Master* mentre gli altri nodi sono detti *Slave* e sono semplici utilizzatori del canale di comunicazione (possono ricevere o trasmettere informazione ma in funzione del controllo del bus fatto dal master). Una rete master/slave in cui ci sono più dispositivi che fungono da master si dice *multi-master*. In tal caso è necessario un sistema di arbitraggio per i risolvere i conflitti che nascono quando più master richiedono il controllo del canale di comunicazione. Tipicamente in queste reti il nodo (master) che prende il controllo del bus, quando inizia una comunicazione, specifica tramite un indirizzo a quale nodo della rete è destinata l'informazione (e.g. bus dati e indirizzi nell'architettura interna di un calcolatore). Esistono anche protocolli detti *producer/consumer* in cui qualsiasi nodo può acquisire momentaneamente il controllo del bus ed iniziare una trasmissione (funge da producer) mentre gli altri nodi si attiveranno in questa fase in ricezione (consumer). Il nodo della rete che trasmette non specifica a quale nodo della rete è destinata l'informazione (ovvero non viene specificato nessun indirizzo) ma specifica da quale interfaccia vengono prodotti i dati. Il CAN è un protocollo di comunicazione di questo tipo.

1.1.2 Interfaccia RS-232

Segnale RS-232

La Figura 2 mostra l'andamento del segnale (trama) che, in una comunicazione tra due dispositivi con protocollo RS-232 a 9600 bits/s 8n2[1] rappresentante il valore binario 00110000.

[1] Il formato del pacchetto trasmesso è indicato da una sigla composta da numeri e cifre, per esempio 8n2 dove la prima cifra indica quanti bit di dati sono trasmessi (8), la prima lettera il tipo di parità (nessuna), la seconda cifra il numero di bit di stop (2).

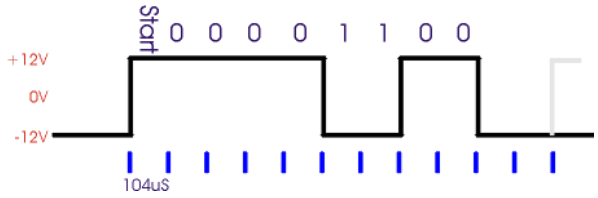


Figura 2: Esempio segnale RS-232

Il segnale, così come nel CAN illustrato di seguito, utilizza per i bit una codifica *NRZ* (*Non-Return-Zero*): il segnale è un treno di impulsi rettangolari di durata T fissata dal bit-rate (T=1/9600=104 µs nell'esempio) e con ampiezza caratterizzata da un valore "alto" pari a circa +12V ed un valore "basso" pari a –12V. Ogni impulso rappresenta un bit. In particolare nella RS-232 un valore alto di tensione rappresenta lo zero logico ed uno basso un uno logico. La linea si trova inizialmente nello stato di riposo con un livello basso di tensione (nessun dato in transito); la prima transizione da livello basso di tensione a livello alto indica l'inizio della trasmissione ("bit di start" della durata di 104 µs). Segue il bit meno significativo (LSB), dopo altri 104 µs il secondo bit, e così via, per otto volte, fino al bit più significativo (MSB). Segue infine un periodo di riposo della linea di almeno 208 µs, cioè due bit di stop e quindi (eventualmente) inizia un nuovo pacchetto di bit. Pertanto nell'esempio considerato vengono trasmessi 11 bit di cui solo 8 rappresentano l'informazione effettivamente utile. Da notare che con la codifica NRZ il numero di commutazioni è inferiore al numero di bit (in Figura 2 abbiamo 4 commutazioni a fronte di 11 bit trasmessi).

Se la trasmissione è più veloce o più lenta, la durata degli impulsi varia (a 1200 kbits/s le transizioni avvengono a multipli di 0,833 ms). Lo standard originale prevede una velocità fino a 20 kbits/s. Uno standard successivo (RS-562) ha portato il limite a 64 kbits/s con i due standard compatibili a bassa velocità. Le interfacce seriali RS-232 nei normali PC in genere superano i 100 kbits/s. Il numero di bit dei dati trasmessi per ogni trama può variare da 5 a 9; è possibile aggiungere un bit di parità per incrementare la robustezza della comunicazione; al termine della comunicazione la linea rimane nello stato di riposo per almeno 1 o 2 bit. Come detto oltre ai bit dei dati viene inserito un bit di parità (opzionale) per verificare la correttezza del dato ricevuto. Esistono cinque tipi di parità:

- *None*: nessun tipo di parità, cioè nessun bit aggiunto
- *Pari* (even): il numero di 1 (incluso il bit di parità) è sempre pari
- *Dispari* (odd): il numero di 0 (incluso il bit di parità) è sempre dispari
- *Mark*: il bit di parità vale sempre 1
- *Space*: il bit di parità vale sempre 0

La tensione di uscita di un trasmettitore RS-232 deve essere compresa in valore assoluto tra 5 e 25 V (valore ridotto a 13 V in alcune revisioni dello standard). A volte le tensioni in uscita sono diminuite a +/- 6V anziché i 12 V dell'esempio per permettere minori emissioni elettromagnetiche. Il ricevitore deve funzionare correttamente con tensioni di ingresso comprese, in valore assoluto, tra 3 V e 25 V. Molti ricevitori commerciali considerano semplicemente una tensione di soglia al valore di +2V (sopra viene riconosciuto un segnale alto, sotto uno basso). Per adattare i segnali utilizzati da circuiti digitali con livelli non direttamente compatibili con la standard RS-232 esistono appositi traslatori di livello che hanno il compito di fornire sia in trasmissione che in ricezione gli opportuni livelli pur non modificando la forma del segnale trasmesso.

Problema della sincronizzazione

Data la possibilità di implementare diverse varianti dello stesso protocollo di comunicazione sia trasmettitore che ricevitore devono accordarsi sul modo di trasmettere i dati prima di iniziare la trasmissione vera e propria. Inoltre è importante garantire il rigoroso rispetto della durata dei singoli bit: infatti non è presente alcun segnale di clock comune tra trasmettitore e ricevitore e l'unico elemento di sincronizzazione è dato dal fronte di salita del bit di start. Poiché il campionamento in ricezione è effettuato di norma al centro di ciascun bit l'errore massimo ammesso è, teoricamente, pari alla durata di mezzo bit. Naturalmente questo limite non tiene conto della possibile difficoltà di riconoscere con precisione il fronte del bit di start (soprattutto su grandi distanze ed in ambiente rumoroso) e della presenza di interferenze intersimboliche tra bit adiacenti.

Universal Asyncrhonous Receiver/Trasmitter

È utile notare che mediamente i diversi processori lavorano con bus paralleli: da 8-bit tipici di microcontrollori per controllo industriale, fino ai 128 bit di processori Very Long Instruction Word dedicati per elaborazioni di segnali multimediali, con 32-bit valore tipico per processori general purpose. Per trasformare il segnale parallelo proveniente dal processore in segnale seriale esistono dei chip detti UART (Universal Asyncrhonous Receiver/Transmitter). In genere vengono gestite dall'hardware tutte le funzioni a basso livello necessarie quali inserimento dei bit di start e di stop, generazione o riconoscimento del bit di parità, generazione di interrupt e spesso è presente un buffer con logica FIFO (First In First Out) che permette di ricevere ed inviare dati anche quando il processore è impegnato. Nei moderni microcontrollori il modulo che fa da UART è spesso integrato sullo stesso chip del core di processamento e pertanto il microcontrollore può comunicare secondo protocolli sia seriali che paralleli.

1.1.3 Interfaccia CAN (Controller Area Network)

CAN (Controller Area Network) è un bus seriale di comunicazione dati progettato per applicazioni real-time: consente a controllori, sensori e attuatori di comunicare l'uno con l'altro ad una velocità fino a 1Mbit/s, tipicamente 500 kbits/s nelle attuali implementazioni (data-rate molto maggiore delle velocità descritte per la RS-232), offrendo anche:

- bassi costi di progettazione e implementazione
- funzionamento in ambienti ostili
- facilità di configurazione e modifica
- rilevamento automatico degli errori di trasmissione.

Nato originariamente per l'industria automobilistica (CAN è stato sviluppato dalla Bosch nel 1986 su richiesta della Mercedes), si è diffuso presto nell'automazione industriale per le sue caratteristiche di robustezza ed affidabilità. Oggi sono disponibili chip di diverse aziende che implementano il protocollo CAN (Philips, Intel, Motorola, Siemens etc.) e sono disponibili diversi integrati che implementano su un unico chip un controllore CAN insieme al core di processamento e una UART.

Caratteristiche generali del protocollo CAN

Il nucleo del protocollo CAN è nel livello data link ovvero nella politica di accesso al mezzo di trasmissione. L'architettura di rete è molto flessibile in quanto possono essere supportate connessioni punto-punto, master/slave o multi-master. Tra le caratteristiche del protocollo abbiamo:

Assenza di indirizzi mittente/destinatario e multicast

I pacchetti trasmessi da un modulo CAN non contengono indirizzi di alcun genere, al loro posto troviamo un identificatore del contenuto del messaggio (e.g. energia, valore efficace di tensione, ecc) unico sull'intera rete. Pertanto, in accordo ad un approccio producer/consumer, il nodo della rete che trasmette non specifica a quale nodo della rete è destinata l'informazione ma specifica da quale interfaccia vengono prodotti i dati. Un nodo ricevitore può verificare il contenuto del messaggio e filtrare i soli pacchetti a cui è interessato (message filtering), ignorando gli altri. Questo modo di operare in cui l'informazione del nodo trasmettitore è accessibile a tutti gli altri nodi della rete è detto Multicast.

Ad esempio in Figura 3 viene rappresentata la situazione di un nodo (nodo 2) che trasmette dati riguardanti e.g. potenza misurata SM1 e, mentre lo SC (nodo 1) accetta il pacchetto, lo SM3 (nodo 3) lo ignora.

Questo approccio di comunicazione (basato sul contenuto dei messaggi trasmessi sulla rete e non sugli indirizzi dei nodi della rete) permette un alto grado di flessibilità e modularità del sistema, consentendo che nuovi nodi che sono solo ricevitori e che hanno bisogno dei soli dati esistenti possano essere aggiunti senza alcuna modifica nè all'hardware nè al software. Nel suo formato standard il CAN supporta identificatori del messaggio a 11 bits (che consentono la codifica di 2048 tipi di messaggi differenti).

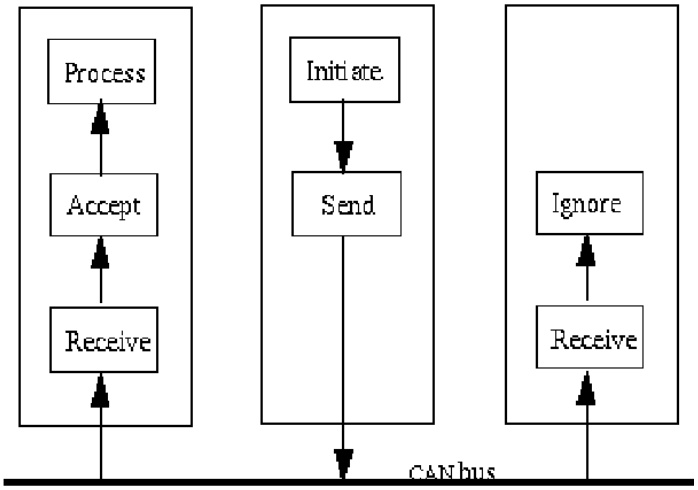


Figura 3: Esempio di message filtering: l'informazione del nodo 2 è ricevuta dal nodo 1 ma ignorata dal nodo 3

Politica di arbitraggio

Ovviamente in una rete in cui vi sono più nodi in grado di generare informazione nasce il problema dell'arbitraggio dell'accesso al bus. Quando il canale (bus) è libero ogni unità connessa può cominciare a trasmettere secondo una politica di trasmissione detta non-destructive bitwise arbitration. Ovvero due o più stazioni che iniziano a trasmettere competono per l'accesso al bus con un valore di priorità determinato proprio dall'identificatore. Quello con il valore numerico più basso vince la competizione per il canale. In pratica l'ID del messaggio definisce una priorità in trasmissione. Il nodo che deve trasmettere il messaggio con priorità maggiore (ID minore: lo 0 vince sull' 1. Al livello logico 0 corrisponde un livello di segnale detto DOMINANT, al livello logico 1 corrisponde un livello di segnale detto RECESSIVE. Una eventuale sovrapposizione di segnali DOMINANT e RECESSIVE sul bus viene interpretato come uno 0 logico) può trasmettere non appena è libero il bus.

In Figura 4 è riportato un esempio di conflitto sull'accesso al bus tra tre nodi che tentano di trasmettere allo stesso istante: in particolare vengono riportati i segnali che tentano di trasmettere ed il segnale effettivo che si legge sul bus. In particolare per i primi tre cicli non c'è conflitto. Al quarto ciclo il nodo 1 perde la competizione e smette di trasmettere perché cerca di imporre un uno logico (recessivo) mentre i nodi 2 e 3 impongono lo zero logico (dominante). In corrispondenza dell'ottavo ciclo il nodo 3 perde la competizione e smette di trasmettere mentre il nodo 2 continua a trasmettere come se fosse stato solo sul bus. Da notare che i nodi perdenti diventano subito ricevitori e non tenteranno una ritrasmissione non prima che il bus sia diventato libero. Questa politica garantisce il determinismo dell'accesso al bus, e l'assenza di periodi di inattività del canale. Anche in questo caso la politica di arbitraggio è basata sul contenuto dei messaggi trasmessi sulla rete e non sugli indirizzi dei nodi della rete.

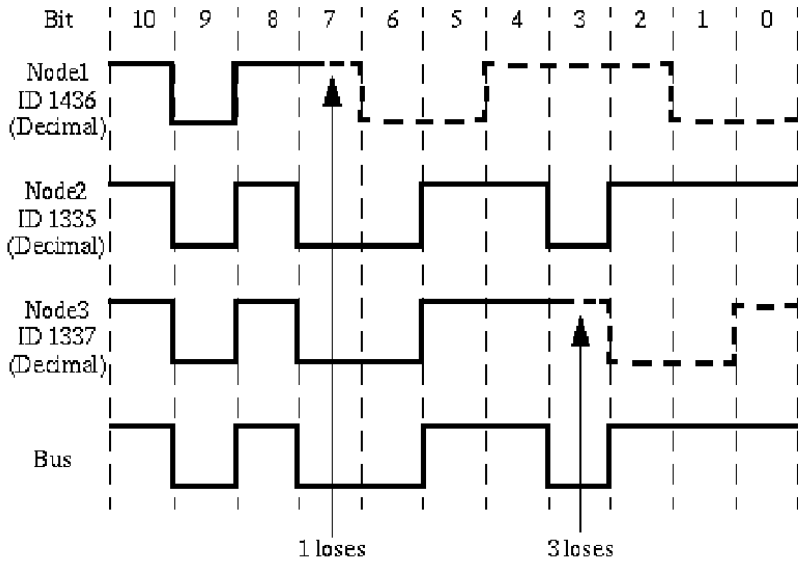


Figura 4: Esempio di arbitraggio nell'accesso al bus CAN

Formati dei messaggi

CAN prevede 4 tipi di messaggi:

1. Data Frame (trasporta dati da un nodo sorgente ad un nodo ricevente);
2. Remote Frame (viene inviato da un nodo per ottenere un DATA FRAME con lo stesso ID);
3. Error Frame (trasmesso da qualsiasi stazione che rilevi un errore sul bus);
4. Overload Frame (serve ad aggiungere ritardi extra)

Ci sono due formati per i Data Frame e Remote Frame:

- Standard CAN (Versione 2.0 A)
- Extended CAN (Versione 2.0 B)

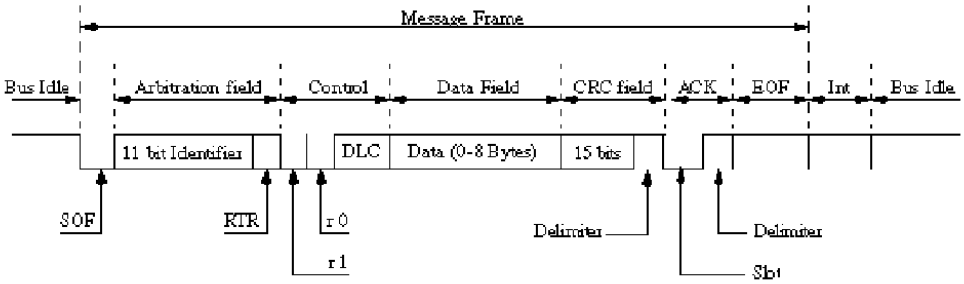


Figura 5: Struttura messaggio nel protocollo CAN 2.0

Con riferimento a Figura 5 lo Standard CAN 2.0 A prevede che il Data Frame sia strutturato come segue:

- Un campo Start Of Frame (SOF). È un bit dominante (0 logico) che indica l'inizio di un message frame. Il rilevamento di un bit dominante quando la linea è in pausa (Bus Idle, sulla linea c'e' un bit 1 recessivo) è quindi interpretato come un SOF. Difatti l'SOF in Figura 5 ha la stessa funzione dello start bit della RS-232.
- Un Arbitration Field, contenente 11 bit di identificatore e il Remote Transmission Request (RTR) bit. Quest'ultimo bit quando è settato a 0 indica che il frame è un Data Frame, se settato a 1 indica che è un Remote Frame. Il Remote Frame è una richiesta da parte di un nodo del Data Frame corrispondente (avente lo stesso identificatore) e non presenta un campo dati. Ovvero un nodo della rete può richiedere dei dati di un tipo specificato (da un certo ID) inviando un Remote Frame (che essendo una richiesta non porta informazione e quindi ha il campo dati Data Field nullo). Il nodo che può fornirli li invia in un Data Frame che ha lo stesso ID del Remote Frame.
- Un Control Field contenente 6 bit: 2 bit riservati per usi futuri e 4 bit di Data Lenght Code (DLC). Questo indica il numero dei byte nel Data Field seguente, esso può variare da 0 a 8 byte (24=16, ma si utilizzano le sole combinazioni "0000" $\rightarrow$ 1000").
- Data Field rappresenta il contenuto informativo vero e proprio.
- Il CRC Field contenente 15 bit di cyclic redundancy check code e un bit recessive come delimitatore. Il CRC difatti rappresenta una evoluzione del concetto di bit di parità ovvero dell'inserzione di bit ridondanti atti ad aumentare la capacità di individuazione di errori nella comunicazione.
- Il campo Ack di 2 bit; il primo è lo Slot Ack che è trasmesso come recessive, ma è sovrascritto con un bit dominante da ogni stazione che riceve correttamente il messaggio; il secondo bit è recessive e svolge il compito di delimitatore.
- Il campo End Of Frame (EOF) che consiste di 7 recessive bit.

- Per separare tra loro frames in sequenza vi è l'Interframe Space che non è un periodo di tempo tra due frame, bensì un insieme di bit ovvero un pacchetto speciale inviato dall'ultimo nodo che ha trasmesso (Overload Frames ed Error Frames non sono preceduti dall'Interframe Space). Il pacchetto ha due (tre) campi:
- Intermission (Int che consta di 3 bits settati a 1; quando sul bus passano i bits di Intermission nessuna stazione può trasmettere Data o Remote Frame: l'unica azione ammessa è la segnalazione di condizioni di overload);
- Bus Idle (di lunghezza arbitraria);
- Suspend Transmission (solo per stazioni in condizioni Error Passive; tale campo, se presente, è posto tra gli altri due. Una stazione Error Passive, dopo l'invio di un messaggio, trasmette l'Intermission seguito da 8 bits a 1 prima di trasmettere un nuovo frame o di liberare il bus. Se durante tale periodo un'altra stazione comincia a trasmettere, questa si arresta).

Da notare che la struttura di un Remote Frame è simile a quella del Data Frame ma Data Field non c'è e RTR bit vale 1 invece di 0.

Lo standard CAN 2.0 B prevede un identificatore di 29 bit, per offrire compatibilità con altri protocolli di comunicazione seriale usati in USA. In particolare l'identificatore è diviso in Base ID lungo 11 bit per garantire la compatibilità con la versione A e in Extension ID di 18 bit. Esistono tre tipi di controllori CAN: i 2.0A che sono capaci di spedire solo messaggi di formato Standard, restituendo errore nel caso ricevano in formato Extended; i 2.0B passive che sono in grado di spedire solo in formato Standard, ma possono ricevere in formato Extended; i 2.0B che possono funzionare in entrambe le modalità.

Rilevamento degli errori

Il processo di segnalazione degli errori in una rete CAN si articola nelle seguenti fasi:

- Un controller CAN rileva un errore (in trasmissione o in ricezione);
- Un Error Frame viene immediatamente trasmesso;
- Il messaggio incriminato viene ignorato da tutti i nodi;
- Viene aggiornato lo stato del controller CAN;
- Il messaggio viene ritrasmesso, eventualmente competendo con altri;

Un errore può essere rilevato in 5 modi, 3 dei quali a livello del messaggio e 2 a livello del singolo bit:

- *Bit Stuffing Error*; normalmente un nodo in trasmissione inserisce dopo 5 bit consecutivi della stesa polarità un bit di polarità opposta; ciò è chiamato bit stuffing. Un nodo che riceve più di 5 bit consecutivi di segno uguale rileverà un errore di questo tipo.
- *Bit Error*; un nodo in trasmissione ascolta sempre il bus per verificare la corrispondenza con ciò che sta trasmettendo: se esso ascolta un bit diverso dal suo verrà segnalato un errore.
- *Checksum Error*; ogni nodo ricevente ricalcola il CRC in base a ciò che ha ricevuto, e se non corrisponde a quello inviato dal mittente viene segnalato un errore.
- *Frame Error*; viene segnalato questo tipo di errore quando vengono violati alcuni campi fissi del pacchetto (bit che devono essere spediti sempre dello stesso tipo).
- *Acknowledgement Error*; se il trasmettitore non rileva alcun riscontro al frame appena inviato.

Un Error Frame è costituito da un *Error Flag* ed un *Error Delimiter*.

L'*Error Flag* è lungo 6 bit dello stesso segno e viola volontariamente la regola dello bit stuffing in modo che tutte le altre stazioni rilevino un errore e spediscano anch'esse un Error Flag. Per questo motivo il campo Error Flag nel pacchetto è di lunghezza variabile (max 12 bit) dato dalla sovrapposizione di tutti gli Error Flag spediti. A seguito c'è un *Error Delimiter* costituito da 8 bit recessive.

Simile all'Error Frame è l'*Overload Frame* poiché anch'esso consiste di un Overload Flag e di un Overload Delimiter. Esso viene generato quando un nodo ricevitore ha bisogno di più tempo per processare i dati correnti prima che altri vengano ricevuti. Un nodo trasmetterà l'overload flag subito dopo l'EOF (End of Frame in Figura 5): in questo modo tutti gli altri nodi diagnosticheranno una condizione di overload e spediranno anch'esse un overload flag. A seguito della sovrapposizione dei flag ci saranno 8 bit recessive di delimiter. Un Overload Frame non richiede la ritrasmissione del frame che ha causato la condizione di overload.

Auto diagnosi dei nodi

CAN offre anche un meccanismo di auto isolamento dei guasti con la possibilità di distinguere tra condizioni di guasto transitorie (e.g. dovute a sbalzi di tensione, condizioni esterne di disturbo), e guasti permanenti (e.g. dovuti a cattive connessioni, cavi rotti). Ogni nodo CAN ha due registri di error count: uno di trasmissione e uno di ricezione. Essi sono inizialmente settati a 0 e vengono incrementati ogni qualvolta si presenta una situazione di errore (+1 per un errore in ricezione, +8 per un errore in trasmissione). A sua volta ogni nodo della rete CAN può trovarsi in tre stati (vedi Figura 6)

- Error Active. Nessuno dei due contatori ha superato il valore di 127. Il nodo è nel pieno delle sue funzionalità e decrementa di 1 i contatori ogni volta che riceve un messaggio andato a buon fine. Quando è in questo stato il nodo che rileva un errore spedisce un Error Flag costituito da 6 bit dominanti in modo da interrompere sempre la trasmissione.
- Error Passive. Almeno uno dei due contatori ha superato 127. Il nodo è ancora in grado di eseguire tutte le sue funzioni, ma è probabile che esso presenti dei disturbi o condizioni di guasto. Per questo quando esso rileva un errore spedisce un Error Flag di 6 bit recessive che vengono interpretati come errore solo se nessuna stazione sta spedendo un suo proprio messaggio (i bit recessive contrariamente vengono sovrascritti).
- Bus Off. Se uno dei contatori supera 255 il nodo si stacca dal bus e non partecipa alla comunicazione lasciando gli altri nodi nella possibilità di continuare a scambiarsi informazioni (autoisolamento). Se ciò accade certamente la stazione presenta un problema permanente che necessita di un intervento esterno per ripristinare il perfetto funzionamento. Alcune implementazioni consentono al nodo di tornare Error Active dopo che esso abbia ricevuto 128 messaggi andati a buon fine, altre necessitano di un reset hardware.

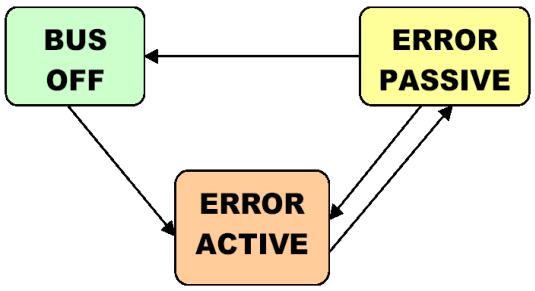


Figura 6: Stati di un nodo CAN

L'alta affidabilità del CAN e il suo successo in applicazioni in cui la sicurezza è un fattore critico (automazione, trasporti, biomedicale) è legata alla capacità di identificare dati corrotti da guasti di trasmissione. La probabilità residua d'errore è una misura statistica e specifica la probabilità che un messaggio sia corrotto ma non diagnosticato tale da nessun nodo della rete. È stato calcolato che su un bus CAN a 1Mbit/s utilizzato al 50%, con lunghezze medie di messaggi di 80 bits si avrebbe una probabilità residua di non individuare un messaggio corrotto di circa $4*10^{-7}$ messaggi/ora (a 8 ore al giorno per 365 giorni l'anno si avrebbe un guasto non diagnosticato ogni 1000 anni).

Livello fisico CAN

Il livello fisico del CAN è stato standardizzato in accordo con la direttiva ISO 11898. Il cavo trasmissivo su cui vengono trasmesse le informazioni in accordo al protocollo seriale fino ad ora analizzato è costituito da una coppia di fili intrecciati (chiamati CAN_H e CAN_L) pilotati in modo differenziale. Se l'informazione è trasmessa in modo differenziale ovvero V= V_{CAN_H} -V_{CAN_L} tutti i disturbi ΔV che si manifestano sulle due linee nello stesso modo scompaiono essendo V= (V_{CAN_H} + ΔV) – (V_{CAN_L} + ΔV)= V_{CAN_H}-V_{CAN_L}. Il cavo può essere sia schermato sia non schermato. L'impedenza che termina il cavo deve essere di 120 Ω. La lunghezza massima del bus dipende dalla velocità di trasmissione usata in accordo ai valori in Figura 7 (bit-rate maggiori sono supportati su connessioni di lunghezza inferiore). In ogni caso il CAN è un bus per connessioni con data-rate massimi di 1 Mbits/s.

Per ridurre la potenza dissipata esiste per i nodi di una rete CAN la possibilità di entrare in una fase di sleep mode (nessuna attività interna, disconnessione dai bus drivers). Il nodo esce dallo sleep mode tramite un segnale di wake-up causato da qualche attività sul bus o da condizioni interne del sistema. Per svegliare nodi in sleep mode esiste un messaggio dedicato di wake up: "11111101111" che deve essere inviato con l'ID più piccolo possibile (infatti più piccolo l'ID del messaggio, maggiore è la sua priorità nel caso di conflitti sul bus).

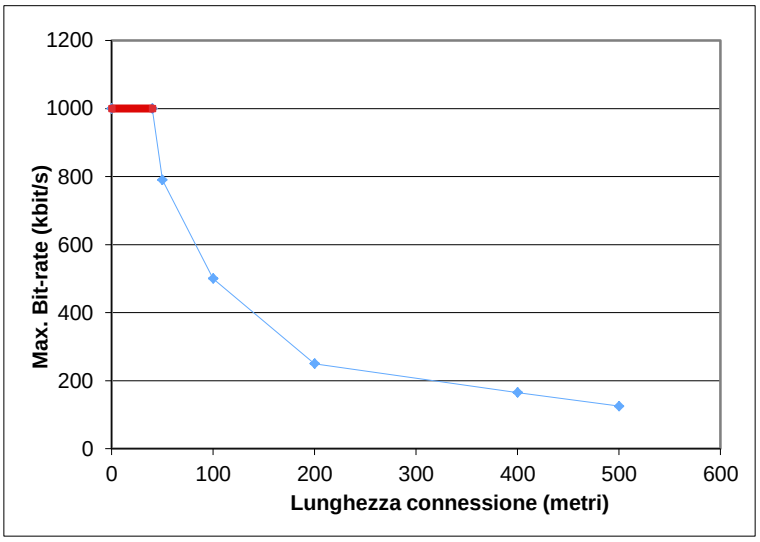


Figura 7: Velocità di trasmissione in funzione della lunghezza del bus

Implementazione di nodi CAN

Il mercato offre un'ampia gamma di dispositivi CAN, più di 50 prodotti diversi (da più di 15 aziende diverse). Nel 2000 sono stati venduti oltre 100 milioni di controllori CAN. In molti casi il controllore CAN è integrato sullo stesso chip del microprocessore.

Ci sono due principali strategie di implementazione per i controller CAN: le sostanziali differenze tra le due sono su come vengono filtrati i messaggi, su come vengono bufferizzati e su come avvengono le risposte ai Remote Frames.

Basic CAN

Il Basic CAN è un controller più economico. Esso possiede dei buffer di ricezione e trasmissione gestiti con politica First-In-First-Out: un messaggio può essere ricevuto su un buffer mentre il microcontrollore sta leggendo su un altro buffer; se arriva un messaggio quando tutti i buffer sono pieni, il più vecchio viene conservato, il che significa che possono andare perse delle informazioni nell'eventualità che il microcontrollore non sia abbastanza veloce. Un messaggio viene inviato scrivendolo in un buffer di trasmissione. I messaggi a cui il nodo è interessato sono filtrati usando due registri che operano sull'identificatore del messaggio: se l'identificatore supera il controllo di tali maschere viene registrato in un buffer. Il filtraggio finale viene fatto a livello software con un extra carico per il microcontrollore.

Il Basic CAN non supporta la risposta automatica ai Remote Frames, ma l'applicazione software dovrà gestirli, garantendo la correttezza della risposta (si ha così un carico extra di lavoro per il processore).

Full CAN

Il Full CAN è un controller più performante e più costoso del Basic. Ha un insieme di buffer chiamati mailboxes i quali al momento dell'inizializzazione del sistema sono settati in trasmissione o in ricezione e ad ognuno viene assegnato un identificatore; questo significa che ad ogni messaggio compete il proprio buffer specifico. Quando viene ricevuto un messaggio vengono controllati tutti i buffer di ricezione cercando quello avente l'identificatore in questione: se viene trovato significa che il messaggio ha un contenuto rilevante per il nodo e viene memorizzato, altrimenti viene scartato in quanto non interessante. In questo modo il filtraggio avviene a livello hardware senza che se ne occupi il processore.

Lo stesso in trasmissione, il messaggio viene memorizzato nel buffer che gli compete; in questo modo può essere attuata anche una politica di selezione del messaggio da trasmettere, favorendo il più prioritario e non una semplice FIFO come nel Basic.

Quando viene ricevuto un Remote Frame, il controller CAN verifica se esiste un buffer di trasmissione con lo stesso identificatore: in caso affermativo, il Data Frame corrispondente viene subito inviato senza chiamare in causa il microcontrollore e snellendo di conseguenza i suoi compiti.

1.2 *Studio ed Individuazione di protocolli efficienti per la cyber-security: scambio dati e memorizzazione sicura delle informazioni attraverso la crittografia.*

Nei sistemi informatici distribuiti sia di monitoraggio che di attuazione, un ruolo di fondamentale importanza è rivestito dal dialogo tra gli attori della comunicazione (es. server locali, server manager, sistemi gestiti da remoto, attuatori, meter, ecc.), che avviene essenzialmente in due modi: invio richieste e ricezione delle notifiche. Ciascun meter, ad esempio, che ha il compito di monitorare costantemente la quantità osservata (es. energia generata), invia notifiche delle misure effettuate al concentratore (server locale) il quale dialoga con il Master Server allo scopo di attuare le politiche di gestione previste per ciascuna specifica circostanza. Affinché nella comunicazione tra server manager e sistemi gestiti sia preservata la segretezza, l'autenticazione e l'integrità dei dati scambiati, è necessario introdurre opportuni protocolli che consentano il soddisfacimento di tali specifiche. Nel seguito vengono analizzati i più noti protocolli crittografici ed, in generale, le principali strategie di attuazione dei meccanismi di sicurezza per comunicazione su reti telematiche. Abbracciando un aspetto leggermente più ampio, si può dire che in generale la crittografia ha come fine ultimo la risoluzione di problemi pratici che coinvolgono la segretezza, l'autenticazione e l'integrità. Gli algoritmi crittografici, però, da soli non servono a nulla e per qualunque applicazione concreta sono necessari dei protocolli. Un protocollo è una serie di passi che coinvolge due o più parti e che serve a portare a termine un compito. Quindi un protocollo ha una sequenza, con un inizio ed una fine, ed ogni passo deve essere eseguito in ordine. Oltre a queste caratteristiche fondamentali, un protocollo deve:

- Essere privo di ambiguità, ogni passo deve essere descritto con chiarezza e non ci devono essere possibilità di incomprensioni.
- Essere completo, quindi deve essere prevista un'azione per qualunque situazione.
- Avere carattere di condivisione ovvero chiunque vi partecipi deve conoscerlo ed accettare di seguirlo.

Un protocollo crittografico è un protocollo che alla base del suo funzionamento implementa la crittografia (fa uso di un qualche algoritmo di crittografia), ma, come vedremo, solitamente il suo obiettivo va al di là della semplice segretezza. In generale, comunque, l'uso della crittografia all'interno di un protocollo serve a prevenire o almeno individuare l'eavesdropping (to eavesdrop significa origliare) e il cheating (to cheat significa ingannare). Nella vita quotidiana ci sono protocolli informali per quasi ogni cosa: giocare a poker, ordinare dei beni telefonicamente, votare alle elezioni, ecc. Al giorno d'oggi però, un numero sempre crescente di relazioni umane non avviene più faccia a faccia, ma su reti di computer, ed i computer, per fare quello che le

persone fanno senza neanche pensarci, hanno bisogno di protocolli formali. Nelle relazioni faccia a faccia la sicurezza e la correttezza sono fondate sulla presenza della persona, ma questo naturalmente non è possibile quando si comunica con reti di computer. Per questo alla base della progettazione di un protocollo si considerano potenziali disonesti, non solo gli utilizzatori, ma anche gli amministratori ed i progettisti della rete stessa.

Protocolli con arbitro, giudice e self-enforcing

Un arbitro è una terza parte disinteressata di cui tutti si fidano. Disinteressata significa che non ha particolari legami con nessuna delle parti coinvolte; fidata significa che tutte le persone che partecipano al protocollo accettano come vero ciò che dice e come corretto ciò che fa. Gli arbitri aiutano a completare protocolli tra parti reciprocamente diffidenti. Nella vita reale esempi comuni di arbitri possono essere avvocati, notai o banche; purtroppo però quello che avviene in questi casi non può essere direttamente trasportato nel mondo dei computer. I principali problemi sono:

- È più facile fidarsi di una terza parte neutrale se la si conosce e se la si può vedere in faccia; inoltre è probabile che due parti reciprocamente diffidenti diffidino anche di un arbitro senza volto che si trova da qualche parte nella rete.
- La rete di computer deve sostenere il costo di mantenimento dell'arbitro (e tutti conosco le parcelle degli avvocati).
- I protocolli con arbitro introducono inevitabilmente dei ritardi.
- L'arbitro deve intervenire in ogni transizione, quindi rappresenta un potenziale collo di bottiglia, soprattutto nelle implementazioni su larga scala. Si può pensare di aumentare il numero di arbitri, ma ciò fa crescere il costo.
- L'arbitro rappresenta un punto molto vulnerabile per chi vuole sovvertire la rete.

Ciononostante i protocolli con arbitro hanno le loro applicazioni pratiche.

Un giudice è una terza parte disinteressata e fidata, come l'arbitro, ma a differenza di questo, non partecipa direttamente ad ogni protocollo: viene chiamato in causa solo in caso di una disputa. Questi protocolli si basano sull'onestà delle parti in causa: se qualcuno però sospetta che ci siano stati degli imbrogli, chiama in causa il giudice. Se il protocollo è progettato correttamente, il giudice è in grado di stabilire con certezza se ci sono stati degli imbrogli e da parte di chi: la possibilità di essere smascherati funziona quindi da deterrente.

Un protocollo self-enforcing è il miglior tipo di protocollo, in quanto è il protocollo stesso a garantire l'imparzialità e la correttezza. Non sono necessari né arbitri, né giudici. Il protocollo è progettato in maniera che non ci possano essere dispute, infatti se una delle parti prova a imbrogliare, l'altra se ne accorge immediatamente e il protocollo termina. In un mondo ideale ogni protocollo sarebbe self-enforcing, ma sfortunatamente non ne esiste uno per ogni situazione.

Attacchi contro i protocolli

Gli attacchi crittografici possono essere condotti contro gli algoritmi crittografici, contro le tecniche usate per implementare gli algoritmi o contro i protocolli crittografici: noi consideriamo i primi due livelli sicuri e analizziamo solo gli attacchi contro i protocolli. Per prima cosa qualcuno non coinvolto nel protocollo (Eve) può ascoltare di nascosto ciò che viene trasmesso (eavesdropping): questo viene chiamato passive attack perché l'aggressore non influisce sul protocollo. Alternativamente un aggressore (Mallory) può cercare di modificare il protocollo a proprio vantaggio: può fingere d'essere qualcun altro, cancellare messaggi, sostituire un messaggio con un altro, ripetere vecchi messaggi, interrompere un canale di comunicazione o alterare informazioni memorizzate in un computer. Questi vengono chiamati active attack perché richiedono un intervento attivo. I passive attacker possono solo raccogliere messaggi e cercare di decifrarli tramite crittoanalisi. Gli active attacker invece possono fare molto di più: possono essere interessati ad acquisire informazioni, a degradare le prestazioni del sistema, a corrompere dati esistenti o ad ottenere accessi non autorizzati. E' anche possibile che l'attacker sia una delle parti coinvolte nel protocollo: in tal caso può mentire durante lo svolgimento del protocollo o non seguirlo affatto. Questo tipo di attacker è chiamato cheater**.** I passive cheater seguono il protocollo, ma cercano di ottenere più informazioni di quanto gli sia concesso. Gli active cheater invece infrangono il protocollo per cercare di imbrogliare. Purtroppo è molto difficile mantenere sicuro un protocollo, soprattutto quando quasi tutte le parti coinvolte sono active cheater, ma talvolta si riesce almeno ad individuare la presenza di active cheating. Naturalmente i protocolli dovrebbero essere sicuri contro il passive cheating.

1.2.1 One-way functions

La nozione di one-way function è fondamentale per la crittografia a chiave pubblica. Sebbene non siano protocolli, le one-way function sono un elemento fondamentale per la maggior parte dei protocolli. Le one-way function sono relativamente facili da calcolare, ma molto più difficili da invertire. In questo contesto "difficile" ha più o meno questo significato: sarebbero necessari

milioni di anni per calcolare *x* nota *f(x)*, anche se fossero utilizzati tutti i computer del mondo. La rottura di un piatto è un buon esempio di one-way function. E' facile frantumare un piatto in centinaia di pezzi, ma non altrettanto rimetterli insieme per ottenere nuovamente un piatto. Tutto ciò è interessante, ma da un punto di vista strettamente matematico non ci sono prove che le one-way function esistano o possano essere costruite. Ciononostante, molte funzioni sembrare essere di questo tipo: possiamo calcolare in maniera efficiente e, almeno finora, non conosciamo nessun modo per invertirle agevolmente. Ad esempio x^2 è facile da calcolare, mentre $x^{1/2}$ è molto più complesso. Nel seguito, ad ogni modo, si supporrà che le one-way function esistano. Le one-way function, comunque, non possono essere usare direttamente: un messaggio cifrato con una one-way function è inutile, in quanto nessuno può decifrarlo. Quindi, per la crittografia a chiave pubblica, abbiamo bisogno di qualcos'altro. Una trapdoor one-way function è un particolare tipo di one-way function, con un trabocchetto segreto. È facile da calcolare in una direzione e difficile nell'altra, ma, se si conosce il segreto, l'inversione non è più un problema.

One-way hash functions

Una one-way hash function ha molti nomi (to hash significa tritare): funzione di compressione, funzione di contrazione, impronta digitale, checksum (somma di controllo) crittografica, verifica di integrità del messaggio, codice di individuazione di manipolazione. Comunque la si chiami, questo tipo di funzione è fondamentale nella crittografia moderna. Le hash function sono usate da molto tempo nel mondo dei computer. Una hash function è una funzione che prende una stringa di ingresso di lunghezza variabile (chiamata pre-image) e la converte in una di uscita (chiamata hash value) di lunghezza fissata (generalmente inferiore). Un semplice esempio di hash function è una funzione che prende la pre-image e restituisce un byte ottenuto dallo XOR di tutti i byte d'ingresso. Una one-way hash function è una hash function che funziona in un solo verso: è facile calcolare il valore di hash da una pre-image, ma è difficile generare una pre-image che faccia ottenere un ben preciso valore di hash. La hash function vista prima non è di tipo one-way, infatti, dato un byte, è banale generare una stringa i cui byte abbiano come XOR il byte desiderato. Con una one-way hash function ciò non è possibile; inoltre una buona one-way hash function è anche collision-free (senza collisioni), cioè è difficile generare due pre-image con lo stesso valore di hash. La hash function è pubblica, la sicurezza risiede nella sua unilateralità. L'uscita dipende dall'ingresso in modo indistinguibile. La variazione di un bit nella pre-image, modifica in media metà dei bit del valore di hash. Dato un valore di hash è computazionalmente impossibile trovare una pre-image che lo generi. Grazie a queste funzioni si può pensare di

prendere le impronte digitali ad un file: se Alice vuole verificare che Bob abbia un certo file (che anche lei ha), ma non vuole che glielo invii, allora gli può chiedere solo il valore di hash. Se Bob invia il corretto valore di hash, Alice può essere praticamente certa che anche lui possiede quel file. Questo genere di tecnica è particolarmente utile nelle transazioni finanziarie, in cui non si vuole che, da qualche parte nella rete, un prelievo di $100 diventi un prelievo di $1.000. Bisogna notare che con questo schema chiunque può verificare la correttezza del valore di hash.

1.2.2 Message Authentication Codes

Un message authentication code (MAC), noto anche come data authentication code (DAC), è una one-way hash function con l'aggiunta di una chiave segreta. Il valore di hash è funzione sia della pre-image, sia della chiave. La teoria è la stessa delle hash function, a parte il fatto che, solo chi conosce la chiave, può verificare il valore di hash. Si può creare un MAC da una hash function o da un algoritmo di crittografia, ma ci sono anche MAC dedicati.

1.2.3 Generazione di sequenze casuali e pseudo-casuali

Sembrerebbe inutile perdere tempo parlando della generazione di sequenze casuali, poiché qualunque compilatore integra già un generatore di questo tipo. Sfortunatamente, però, tali generatori di numeri casuali non sono affatto sicuri per la crittografia e, probabilmente, non sono nemmeno casuali. I generatori di numeri casuali non sono veramente casuali perché spesso non è necessario che lo siano. La maggior parte delle applicazioni comuni, come i giochi per computer, ha bisogno di così pochi numeri casuali, che non si accorge di eventuali difetti nel generatore. Al contrario, la crittografia è estremamente sensibile alle proprietà del generatore. Usando un generatore di numeri casuali scadente, si ottengo bizzarre correlazioni e strani risultati. Se la sicurezza dipende dal generatore di numeri casuali, bizzarre correlazioni e strani risultati sono l'ultima cosa che si desidera. Il problema è che un generatore di numeri casuali, in realtà, non produce numeri casuali. Probabilmente, non produce niente di lontanamente simile ad una sequenza di numeri casuali, anche perché è impossibile produrre qualcosa di veramente casuale su un computer. I computer, infatti, sono oggetti deterministici: dei dati entrano da una parte, all'interno avvengono delle operazioni completamente prevedibili e altri dati escono dall'altra parte. Se si inseriscono gli stessi dati in due diverse occasioni, entrambe le volte si ottiene lo stesso risultato. Se si inseriscono gli stessi dati in due computer identici, i risultati sono identici. Un computer può essere solo in un numero finito di stati (un numero molto grande, ma comunque finito) e, ciò che ne esce, sarà sempre una funzione deterministica di ciò che è stato

inserito e dello stato corrente. Ciò significa che ogni generatore di numeri casuali su un computer è, per definizione, periodico. Tutto ciò che è periodico è, per definizione, prevedibile. E se qualcosa è prevedibile, non può essere casuale. Un vero generatore di numeri casuali ha bisogno di un ingresso casuale, cosa che un computer non può fornire.

Sequenze pseudo-casuali

La cosa migliore che un computer può produrre è un generatore di sequenze pseudo-casuali. In modo informale, si può definire pseudo-casuale una sequenza che sembra casuale. Il periodo della sequenza dovrebbe essere abbastanza lungo da garantire che una sequenza finita di lunghezza ragionevole (che poi è quella effettivamente utilizzata) sia non periodica. Se si ha bisogno di un miliardo di bit casuali, non si può usare un generatore che ripete la stessa sequenza ogni sedicimila bit. In generale, queste sottosequenze non periodiche dovrebbero essere indistinguibili dalle sequenze casuali. Per esempio, dovrebbero avere all'incirca lo stesso numero di uni e zeri; circa metà delle serie (sequenze dello stesso bit) dovrebbero avere lunghezza uno, un quarto lunghezza due, un ottavo lunghezza tre, e così via. Non dovrebbero essere possibile comprimerle. Le distribuzioni delle lunghezze delle serie dovrebbero essere le stesse per gli uni e per gli zeri. Per i nostri scopi, un generatore di sequenze è pseudo-casuale se ha la proprietà di sembrare casuale. Questo significa che supera tutti i test statistici di casualità che si possono trovare. Sono stati fatti grossi sforzi per produrre delle buone sequenze pseudo-casuali su computer. Tutti i generatori sono periodici, ma con potenziali periodi di 2^{256} bit ed oltre, possono essere usati nella maggior parte delle applicazioni. Resta il problema delle bizzarre correlazioni e degli strani risultati. Ogni generatore di numeri pseudo-casuali ha questi problemi, se usato in un certo modo: questo è ciò che il crittanalista userà per attaccare il sistema.

Sequenze pseudo-casuali crittograficamente sicure

Per le applicazioni crittografiche la casualità statistica non è sufficiente. Una sequenza pseudocasuale, per essere crittograficamente sicura, deve avere anche la seguente proprietà: essere imprevedibile. Deve essere computazionalmente impossibile prevedere il prossimo bit casuale, data una completa conoscenza dei bit precedenti e dell'algoritmo che genera la sequenza. Per concludere ci chiediamo se esistano delle sequenze veramente casuali. Filosofia a parte, per noi, un generatore di sequenze è veramente casuale se ha anche questa proprietà: Non può essere riprodotto. Se si usa il generatore di sequenze due volte con lo stesso ingresso, i due risultati saranno completamente correlati. Alla fine il problema rimane sempre quello di determinare se una data sequenza è veramente casuale.

1.2.4 Protocolli che utilizzano la crittografia simmetrica

Se due persone vogliono comunicare in sicurezza, naturalmente cifrano i messaggi, ma il protocollo non è solo questo; vediamo cosa deve fare Alice per inviare un messaggio cifrato a Bob:

1. Alice e Bob si accordano su un cryptosystem (sistema crittografico).
2. Alice e Bob si accordano su una chiave.
3. Alice prende il plaintext message (messaggio in chiaro) e lo cifra usando l'algoritmo crittografico e la chiave. In questo modo ottiene un ciphertext message (messaggio cifrato).
4. Alice invia il ciphertext message a Bob.
5. Bob decifra il ciphertext message con lo stesso algoritmo e la stessa chiave, quindi lo legge.

Eve, che si trova tra Alice e Bob, può sferrare un ciphertext-only attack, ma ci sono algoritmi che, per quanto ne sappiamo, resistono a qualunque potenza di calcolo realisticamente a disposizione di Eve. Naturalmente Eve non è stupida, quindi può cercare di ascoltare anche i passi 1 e 2 del protocollo: in tal caso, quando intercetta il ciphertext message, non fa altro che decifrarlo e leggerlo. In un buon cryptosystem la sicurezza dipende esclusivamente dalla conoscenza della chiave, mentre l'algoritmo può anche essere reso pubblico. Per questo la gestione delle chiavi riveste un ruolo molto importante. La chiave deve restare segreta prima, durante e dopo l'esecuzione del protocollo, o comunque fino a quando si vuole che il messaggio rimanga segreto. Come conseguenza di quanto detto, il passo 1 può essere eseguito in pubblico, mentre il passo 2 deve essere necessariamente realizzato in segreto. Mallory, un active attacker, può fare parecchie altre cose. Può provare ad interrompere le comunicazioni al passo 4, impedendo ad Alice di comunicare con Bob. Può intercettare i messaggi di Alice, sostituendoli con i propri. Nel caso conosca la chiave (intercettandola al passo 2 o violando il cryptosystem), può cifrare i propri messaggi e inviarli a Bob: a questo punto Bob penserà che i messaggi arrivino da Alice. Oppure può creare un nuovo messaggio anche senza conoscere la chiave: in questo caso Bob, decifrando il messaggio, otterrà un discorso senza senso e penserà che Alice (o la rete) abbia dei seri problemi. Naturalmente anche gli stessi Bob e Alice possono violare il protocollo, ma la crittografia simmetrica assume che i due si fidino l'uno dell'altro. Per concludere i cryptosystem simmetrici hanno i seguenti problemi:

- Le chiavi devono essere distribuite segretamente. Questo è un problema soprattutto nelle comunicazioni a livello mondiale. Spesso le chiavi sono consegnate a mano da dei corrieri.
- Se una chiave è compromessa (rubata, indovinata, estorta, comprata,ecc.), Eve può decifrare tutti i messaggi. Può anche fingere di essere una delle parti e produrre falsi messaggi per ingannare le altri parti.
- Assumendo che sia usata una diversa chiave per ogni coppia di utenti in una rete, il numero totale di chiavi cresce rapidamente al crescere del numero di utenti: in generale una rete con n utenti richiede $n(n-1)/2$ chiavi.

1.2.5 Protocolli che utilizzano la crittografia a chiave pubblica

Si può pensare ad un algoritmo simmetrico come ad una cassaforte. La chiave è la combinazione. Qualcuno che conosce la combinazione può aprire la cassaforte, metterci un documento e quindi richiuderla. Qualcun altro che conosce la combinazione può riaprire la cassaforte e prendere il documento. Chi non conosce la combinazione deve invece imparare a scassinare casseforti. Nel 1976 Diffie e Hellman cambiarono per sempre il mondo della crittografia, descrivendo la crittografia a chiave pubblica (la NSA ha sostenuto di conoscere il concetto dal 1966, ma non ne ha fornito prova). Questo rivoluzionario sistema crittografico usa due chiavi, una pubblica ed una privata. E' computazionalmente difficile dedurre la chiave privata da quella pubblica. Chiunque, usando la chiave pubblica, può cifrare un messaggio, ma non decifrarlo. Solo chi possiede la chiave privata può decifrare. E' come se la cassaforte fosse diventata una cassetta postale: chiunque può imbucare una lettera, ma solo chi ha la chiave della cassetta può ritirare la posta. Matematicamente il processo è basato sulle trapdoor one-way function: la cifratura è la direzione facile e la chiave pubblica rappresenta le istruzioni per la cifratura, così chiunque può cifrare un messaggio. La decifratura è la direzione difficile, talmente difficile che, anche chi possiede dei computer Cray e migliaia (se non milioni) di anni, non può realizzarla, se non conosce il segreto. Il segreto, o trapdoor, è la chiave privata: se la si conosce, la decifratura è facile come la cifratura. Questo è lo schema con cui Alice invia un messaggio a Bob usando la crittografia a chiave pubblica:

1. Alice e Bob si accordano su un cryptosystem a chiave pubblica.
2. Bob invia ad Alice la propria chiave pubblica.
3. Alice cifra il proprio messaggio usando la chiave pubblica di Bob e glielo invia.
4. Bob decifra il messaggio di Alice usando la propria chiave privata.

Si noti come la crittografia a chiave pubblica risolva il problema di gestione delle chiavi della crittografia simmetrica. Prima Alice e Bob dovevano accordarsi segretamente su una chiave, ora invece possono scambiarsi messaggi segreti senza accordi preventivi. Solitamente gli utenti di una rete si accordano su un comune cryptosystem a chiave pubblica. Ogni utente ha la propria chiave pubblica e la propria chiave privata, e le chiavi pubbliche sono pubblicate in un database. Ora il protocollo è ancora più semplice:

1. Alice ottiene la chiave pubblica di Bob dal database.
2. Alice cifra il proprio messaggio usando la chiave pubblica di Bob e glielo invia.
3. Bob decifra il messaggio di Alice usando la propria chiave privata.

Nel primo protocollo, Bob deve inviare ad Alice la propria chiave pubblica prima che lei possa inviargli un messaggio. Il secondo invece è più simile alla posta tradizionale, infatti Bob non è coinvolto nel protocollo finché non vuole leggere il messaggio.

1.2.6 Sistemi crittografici ibridi

Nel mondo reale gli algoritmi a chiave pubblica non sostituiscono quelli a chiave privata. Infatti non sono usati per cifrare i messaggi, bensì per cifrare le chiave. Questo, principalmente, per due ragioni:

1. Gli algoritmi a chiave pubblica sono lenti, almeno 1.000 volte più lenti di quelli a chiave privata.
2. I cryptosystem a chiave pubblica sono vulnerabili ai chosen-plaintext attack.

Se $C=E(P)$, quando P è un plaintext scelto da un insieme di n elementi, un crittanalista deve solo cifrare tutti gli n possibili plaintext e confrontare i risultati con C. In questo modo non potrà ottenere la chiave privata, ma potrà determinare P. Un chosen-plaintext attack può essere particolarmente efficace se ci sono pochi messaggi cifrati. Per esempio, se P è una cifra minore di 1.000.000, il crittanalista prova un milione di cifre diverse. Questo attacco può essere molto efficace anche quando P non è così ben definito. In certi casi, infatti, anche solo sapere che un ciphertext non corrisponde ad un particolare plaintext, può essere un'informazione utile. I sistemi simmetrici non sono vulnerabili a questo attacco, perché un crittanalista non può effettuare cifrature di prova con una chiave sconosciuta. Nella maggior parte delle implementazioni pratiche la crittografia a chiave pubblica è usata per distribuire in modo sicuro le session key. Tale chiavi vengono poi usate per scambiarsi messaggi tramite algoritmi simmetrici. Questo è spesso chiamato hybrid cryptosystem:

1. Bob invia ad Alice la propria chiave pubblica.
2. Alice genera una session key casuale, *K*, la cifra usando la chiave pubblica di Bob e la invia a Bob.

   EB(K)

3. Bob decifra il messaggio di Alice usando la propria chiave privata per recuperare la session key.

   DB(EB(K))=K;

4. Entrambi cifrano le loro comunicazioni usando la stessa session key.

Nota: *EB* deriva dall'inglese to encrypt (cifrare); *DB* deriva dall'inglese to decrypt (decifrare); il pedice *B* dice che si stanno usando le chiavi (pubblica e privata) di Bob. L'uso della crittografia a chiave pubblica per la distribuzione delle chiavi risolve un importante problema di gestione delle chiavi. Con la crittografia simmetrica, la chiave esiste già prima di essere usata: se Eve riesce a metterci le mani sopra, può decifrare tutti i messaggi. La session key, invece, è creata quando serve e distrutta quando non serve più. Questo riduce drasticamente il rischio di compromettere la session key.

1.2.7 Firme Digitali

Le firme scritte a mano sono usate da molto tempo come prova di paternità di un documento, o almeno di accordo con esso. La firma è convincente per vari motivi:

1. La firma è autentica, convince il destinatario del documento che l'autore ha deliberatamente firmato il documento.
2. La firma non si può falsificare, è la prova che il firmatario, e nessun altro, ha deliberatamente firmato il documento.
3. La firma non è riutilizzabile, è parte del documento e nessuno può spostarla su un altro documento.
4. Il documento firmato è inalterabile.
5. La firma non può essere ripudiata,ovvero il firmatario non può sostenere, in seguito, di non aver firmato.

In realtà, nessuna di queste cose è proprio vera, ma siamo pronti ad accettare questa situazione perché imbrogliare non è facile e si rischia di essere scoperti. Il nostro obiettivo è realizzare la firma tramite computer, ma ci sono dei problemi.

Tutti gli algoritmi di firma digitale sono a chiave pubblica, con una informazione segreta per firmare i documenti ed una informazione pubblica per verificare la firma. Talvolta il processo di firma è detto cifratura con chiave privata e il processo di verifica è detto decifratura con chiave pubblica. Tutto ciò è ingannevole ed è vero per il solo algoritmo RSA. In generale, ci riferiremo ai processi di firma e di verifica senza alcun dettaglio sugli algoritmi coinvolti. La firma di un messaggio con la chiave privata *K* è: S*K(M)* La verifica di una firma con la corrispondente chiave pubblica è: *VK(M)* La stringa di bit unita al documento quando è firmato, sarà chiamata firma digitale, o semplicemente firma. L'intero protocollo, con cui il destinatario è convinto dell'identità del mittente e dell'integrità del messaggio, e chiamato autenticazione.

1.2.8 Scambio delle chiavi

Una tecnica crittografica usata comunemente consiste nell'usare una diversa chiave per ogni conversazione. In tal caso la chiave è chiamata session key, perché è usata per una sola sessione. L'uso di una session key ha molti vantaggi, ma anche un grande problema, cioè far pervenire di volta in volta la chiave alle parti.

Crittografia simmetrica

Questo protocollo prevede che Alice e Bob, utilizzatori di una rete, condividano una chiave segreta (una per Alice e una per Bob) con il Key Distribution Center (KDC), che in questo caso è rappresentato da Trent, una entità di cui tutti si fidano. Tali chiavi devono essere già al loro posto prima dell'inizio del protocollo e, anche se questo è in realtà un grosso problema, ora non ce ne preoccupiamo.

1. Alice chiama Trent e richiede una session key Kab per comunicare con Bob.

2. Trent genera una session key Kab casuale. Ne cifra due copie: una con la chiave di Alice, l'altra con la chiave di Bob.

EKbt(Kab) e EKat(Kab)

3. Trent invia entrambe le chiavi ad Alice.

4. Alice decifra la propria copia della session key.

   DKat [EKat(Kab)] = Kab

5. Alice invia a Bob la sua (di lui) copia della session key.

   EKtb(Kab)

6. Bob decifra la propria copia della session key.

   DKbt [EKtb(Kab)] = Kab

Alice e Bob usano questa session key per comunicare con sicurezza. Questo protocollo si basa sull'assoluta affidabilità di Trent: se Mallory corrompe Trent, tutta la rete è compromessa. Inoltre in questo sistema Trent rappresenta un potenziale collo di bottiglia, in quanto è coinvolto in ogni scambio di chiavi.

Crittografia a chiave pubblica (o a chiave asimmetrica)

Alice e Bob usano la crittografia a chiave pubblica per accordarsi su una session key, che poi usano per cifrare i dati. In pratica, le chiavi pubbliche firmate degli utenti spesso si trovano in un database, per cui Alice può inviare un messaggio sicuro a Bob, anche se lui non la conosce.

1. Alice ottiene la chiave pubblica di Bob dal KDC.

   Get eB

2. Alice genera una session key casuale Kab.

3. Alice cifra Kab usando la chiave pubblica di Bob e la invia a Bob.

   EB (Kab) → Bob

4. Bob decifra il messaggio di Alice usando la propria chiave privata.

   DB [EB (Kab)] = Kab

5. Entrambi cifrano le loro comunicazioni usando la stessa session key.

Pregi - Non c'è bisogno della presenza di Trust

Difetti - Costoso computazionalmente

Man-in-the-middle attack

Eve può solo provare a forzare l'algoritmo a chiave pubblica o tentare un ciphertext-only attack sul messaggio cifrato. Mallory, invece, può fare molto di più: può modificare i messaggi, può cancellarli, può crearne di nuovi; può imitare Bob quando parla con Alice, e viceversa. Vediamo come funziona l'attacco:

1. Alice invia a Bob la propria chiave pubblica. Mallory intercetta questa chiave e invia a Bob la propria chiave pubblica.

2. Bob invia ad Alice la propria chiave pubblica. Mallory intercetta questa chiave e invia ad Alice la propria chiave pubblica.

3. Quando Alice invia un messaggio a Bob, cifrato con la chiave pubblica "di Bob", Mallory lo intercetta. Siccome il messaggio è cifrato con la sua chiave pubblica, lo decifra con la sua chiave privata, lo cifra con la chiave pubblica di Bob e lo invia a Bob.

4. Quando Bob invia un messaggio ad Alice, cifrato con la chiave pubblica "di Alice", Mallory lo intercetta. Siccome il messaggio è cifrato con la sua chiave pubblica, lo decifra con la sua chiave privata, lo cifra con la chiave pubblica di Alice e lo invia ad Alice.

Questo attacco funziona anche quando le chiavi pubbliche di Alice e Bob sono memorizzate in un database. Mallory può intercettare la richiesta di Alice al database e sostituire la propria chiave pubblica a quella di Bob. Naturalmente può fare lo stesso anche con Bob. O, ancora meglio, può introdursi clandestinamente nel database e sostituire la propria chiave sia a quella di Alice che a quella di Bob. Quest'attacco funziona perché Alice e Bob non hanno modo di verificare se stanno realmente parlando fra loro due.

Protocollo interlock

Il protocollo interlock (sincronizzazione) ha buone probabilità di sventare il man-in-themiddle attack. Ecco come funziona:

1. Alice invia a Bob la propria chiave pubblica.

eA → Bob

2. Bob invia ad Alice la propria chiave pubblica.

eB → Alice

3. Alice cifra il proprio messaggio usando la chiave pubblica di Bob, quindi invia metà del messaggio cifrato a Bob.

[EB (Mab)]/2 → Bob

4. Bob cifra il proprio messaggio usando la chiave pubblica di Alice, quindi invia metà del messaggio cifrato ad Alice.

[EA (Mab)]/2 → Alice

5. Alice invia l'altra metà del messaggio cifrato a Bob.

[EB (Mab)]/2 → Bob

6. Bob riunisce le due metà del messaggio di Alice e lo decifra con la propria chiave privata.

DB [EB (Mab)]

7. Bob invia l'altra metà del proprio messaggio ad Alice.

[EA (Mab)]/2 → Alice

8. Alice riunisce le due metà del messaggio di Bob e lo decifra con la propria chiave privata.

DA [EA (Mab)]

L'importante è che mezzo messaggio da solo è inutile. Bob non può leggere niente del messaggio di Alice fino al passo 6; Alice non può leggere niente del messaggio di Bob fino al punto 8. Ci sono vari sistemi per fare ciò:

- Se l'algoritmo di cifratura è un algoritmo a blocchi, la metà di ogni blocco può essere spedita in ognuna delle due metà del messaggio.
- La decifratura del messaggio potrebbe dipendere da un vettore di inizializzazione, da inviare con la seconda metà del messaggio.
- La prima metà del messaggio potrebbe essere una one-way hash function del messaggio cifrato e il messaggio cifrato vero e proprio potrebbe essere la seconda metà.

Per capire come ciò provochi dei problemi a Mallory, esaminiamo il suo tentativo di sovvertire il protocollo. Egli può ancora sostituire la propria chiave pubblica a quelle di Alice e Bob, ma ora, quando intercetta la prima metà del messaggio di Alice, non può decifrarlo e ricifrarlo, quindi

non gli resta che inventare un messaggio completamente nuovo e inviarne metà a Bob. Quando intercetta la prima metà del messaggio di Bob, ha lo stesso problema. Quando poi intercetta le seconde metà, non può più modificare i messaggi che ha inventato in precedenza. La conversazione tra Alice e Bob sarà quindi completamente diversa. Mallory potrebbe anche andare avanti in questo modo. Se conosce abbastanza bene Alice e Bob da imitarli entrambi in una conversazione tra loro, i due potrebbero non accorgersi di venire ingannati. Ma naturalmente questo è molto più difficile di quello che poteva fare prima.

Scambio delle chiavi con firme digitali

L'uso delle firme digitali in un protocollo per lo scambio di session key, elude il man-in-themiddle attack. Trent firma sia la chiave pubblica di Alice che quella di Bob. Le chiavi firmate includono un certificato di proprietà firmato. Quando Alice e Bob ricevono le chiavi, verificano entrambi la firma di Trent. Ora sanno a chi appartiene la chiave pubblica e il protocollo di scambio delle chiavi può procedere. Mallory ha dei seri problemi. Le uniche cose che può fare sono ascoltare il traffico cifrato o distruggere le linee di comunicazione. Questo protocollo usa Trent, ma il rischio di compromettere il KDC è minore che nel primo caso. Se Mallory corrompe Trent, tutto ciò che ottiene è la chiave privata di Trent. Questa chiave gli permette di firmare nuove chiavi; questo però non gli permette direttamente di decifrare delle session key o di leggere dei messaggi cifrati. Per leggere le comunicazioni, Mallory deve impersonare un utente della rete ed effettuare un man-in-the-middle attack. Quest'attacco funziona, ma va ricordato che Mallory deve essere in grado di intercettare e modificare i messaggi. In alcune reti ciò è molto difficile: su un canale broadcast è quasi impossibile sostituire un messaggio (sebbene sia relativamente facile disturbare le comunicazioni per renderle incomprensibili). Sulle reti di computer, invece, questi tipi di attacco diventano ogni giorno più facili (IP spoofing, attacchi ai router, e così via).

1.2.9 Servizi di timestamping (o di certificazione del tempo)

In molte situazioni c'è la necessità di provare che un documento esisteva già in una certa data. Si pensi, ad esempio, ad una controversia su un brevetto: la parte che esibisce la copia meno recente del lavoro, vince la causa. Con i documenti cartacei, i notai possono firmare e gli avvocati tutelare le copie. Se sorge una disputa, il notaio o l'avvocato testimoniano che il documento esisteva già in una certa data. Nel mondo digitale, le cose sono molto più complicate. Non è possibile individuare segni di manomissione in un documento digitale, in quanto può essere copiato e modificato a piacere senza che nessuno se n'accorga. E' banale modificare la data in un

documento per computer. Quello che si vuole ottenere è un protocollo di timestamping (letteralmente "timbro datario") digitale con le seguenti proprietà:

- I dati stessi devono essere timbrati, indipendentemente dal mezzo fisico su cui risiedono.
- Deve essere impossibile cambiare anche un solo bit del documento, senza che la modifica sia evidente.
- Deve essere impossibile eseguire il timestamping di un documento con una data e un'ora diverse dalle attuali.

Una prima soluzione è un protocollo con la presenza di un arbitro, ad esempio Trent, che è un fidato servizio di timestamping, e di Alice, che desidera eseguire il timestamping su di un documento.

1. Alice trasmette una copia del documento a Trent.

2. Trent memorizza la data e l'ora in cui ha ricevuto il documento e tiene in custodia una copia del documento.

Ora, se qualcuno mette in dubbio il momento di creazione del documento, Alice deve semplicemente chiamare in causa Trent. Egli mostrerà la propria copia del documento e verificherà data e ora. Questo protocollo funziona, ma ha degli ovvi problemi. Per prima cosa non c'è privacy. Alice deve fornire una copia del documento a Trent. Chiunque in ascolto sul canale di comunicazione può leggerlo. Alice può cifrarlo, ma in ogni caso una copia del documento deve restare nel database (non si sa quanto affidabile) di Trent. Il secondo problema, è che il database deve essere enorme; inoltre, la larghezza di banda necessaria a spedire file di grandi dimensioni a Trent può non essere disponibile. Il terzo problema riguarda potenziali errori. Un errore di trasmissione o la rottura del computer di Trent, possono annullare completamente la validità del timestamp sul documento di Alice. Infine, può non esistere una persona abbastanza onesta da gestire il servizio di timestamping.

Le **one-way hash function** e le firme digitali possono risolvere facilmente la maggior parte dei problemi:

1. Alice calcola il valore di one-way hash del documento e trasmette il valore di hash a Trent.

   H (M) → Trent

2. Trent appone al valore di hash la data e l'ora in cui lo ha ricevuto e quindi firma digitalmente il risultato e invia il valore di hash (con timestamp) firmato ad Alice.

DTs [H(M) + T] → Alice

Volendo verificare la data conoscendo il messaggio M, servirà la chiave pubblica di Trent,

ETs: ETs { DTs [H(M) + T] } = H(M) + T

In tale modo si verifica che T coincida. Per verificare il messaggio M basterà farne la hash e confrontare il risultato con la hash del messaggio depositato. La data T si può modificare solo conoscendo la chiave privata di Trent quindi Alice e Trent possono sempre agire in collusione per produrre timestamp falsi e comunque Trent potrebbe sempre venire corrotto da una terza parte.

Un modo per risolvere questo problema consiste nel collegare il timestamp di Alice con quelli precedentemente generati da Trent (Protocollo di collegamento o di linking). Tali timestamp saranno molto probabilmente generati per persone diverse da Alice. Siccome l'ordine in cui Trent riceve le diverse richieste di timestamp non è noto a priori, il timestamp di Alice deve essere avvenuto dopo quello precedente. E siccome la richiesta che è arrivata dopo è collegata con il timestamp di Alice, allora la richiesta di Alice deve essere avvenuta prima. Ciò incastra la richiesta di Alice in un preciso slot temporale. Se è il nome di Alice, il valore di hash di cui Alice vuole il timestamp è il timestamp precedente è *Tn-1*, allora il protocollo è:

1. Alice invia a Trent *Hn* e *A*.

2. Trent invia ad Alice:

Tn = SK (n,A,Hn,tn,In-1,Hn-1,Tn-1,Ln)

dove *Ln* è composto dalle seguenti informazioni di collegamento:

Ln = H (In-1,Hn-1,Tn-1,Ln-1)

SK indica che il messaggio è firmato con la chiave privata di Trent. Il nome di Alice la identifica come l'origine della richiesta. Il parametro *n* indica l'ordine della richiesta: questo è il timestamp *n*-esimo emesso da Trent. Il parametro *tn* è il riferimento temporale. L'informazione addizionale è costituita dall'identificativo, dal valore di hash originale, dal

riferimento temporale e dal timestamp con hash del precedente documento certificato da Trent.

3. Dopo avere timbrato il documento successivo, Trent invia ad Alice l'identificativo di chi lo ha richiesto: *In+1*.

Se qualcuno mette in dubbio il timestamp di Alice, lei deve solo contattare gli autori del documento precedente e di quello successivo: *In-1* and *In+1*. Se i loro documenti sono messi in discussione, possono mettersi in contatto con *In-2* and *In+2*, e così via. Tutti possono dimostrare che il loro documento è stato timbrato dopo il precedente e prima del successivo. Questo protocollo rende molto difficili eventuali collusioni tra Alice e Trent. L'unica modo per violare questo sistema consiste nel creare una catena fittizia di documenti, sia prima che dopo quello di Alice, abbastanza lunga da esaurire la pazienza di chiunque stia mettendo in dubbio il timestamp.

Le persone muoiono; i timestamp vanno persi. Possono succedere tante cose che rendono impossibile per Alice ottenere una copia del timestamp di *In-1*. Questo problema può essere alleviato inserendo nel timestamp di Alice i timestamp delle 10 persone precedenti e, poi, inviandole gli identificativi delle 10 persone successive. Alice ha così una probabilità molto maggiore di trovare qualcuno che abbia ancora il proprio timestamp. Il seguente - Protocollo distribuito - è un protocollo del tipo appena visto, che, però, fa a meno di Trent:

1. Usando *Hn* come ingresso, Alice genera una sequenza di valori casuali, per mezzo di un generatore di numeri pseudo-casuali crittograficamente sicuro: *V1, V2, V3,… Vk*

2. Alice interpreta questi valori come gli identificativi di altre persone, quindi invia *Hn* ad ognuna di queste persone.

3. Ognuna di queste persone, allega la data e l'ora al valore di hash, firma il tutto e lo rispedisce ad Alice.

4. Alice memorizza tutte le firme come timestamp.

Il generatore di numeri pseudo-casuali crittograficamente sicuro del passo 1 impedisce ad Alice di scegliere dei controllori corrotti. Anche se prova a fare dei banali cambiamenti nel documento per ottenere un gruppo di identificativi corrotti, le probabilità di riuscita sono trascurabili. Questo protocollo funziona perché, per Alice, l'unico modo di falsificare un timestamp sarebbe

convincere tutte le *k* persone a collaborare. Siccome queste sono scelte casualmente, le probabilità sono tutte contro Alice.

1.2.10 Autenticazione

Quando Alice utilizza un computer host, questo in qualche modo deve riconoscerla. Tradizionalmente il problema è risolto utilizzando delle password: Alice inserisce la propria password ed il computer host conferma che è corretta.

In realtà non è necessario che il computer host conosca le password: è sufficiente che sappia distinguere quelle valide da quelle non valide. Ciò è facile se si utilizzano le one-way function. Invece di memorizzare le password, il computer host ne memorizza le rispettive one-way function. 1. Alice invia al computer host la propria password. 2. Il computer host calcola la one-way function per la password. 3. Il computer host confronta il risultato della one-way function con il valore memorizzato. Siccome il computer host non deve più conservare l'elenco delle password valide, diminuisce il rischio che qualcuno s'intrufoli nel sistema e le rubi. La lista di password cifrate dalla one-way function è inutile, perché la funzione non può essere invertita per recuperare le password.

Un file di password cifrate con una one-way function è ancora vulnerabile. Mallory compila una lista con le password più comuni, con 1.000.000 di elementi. A questo punto usa la oneway function su tutti gli elementi della lista e memorizza i risultati. In seguito, Mallory ruba un file di password cifrate e lo confronta con il suo file, trovando le eventuali corrispondenze. Questo è un dictionary attack, ed è sorprendentemente efficace. Il salt (letteralmente sale, ma forse rende più l'idea la parola condimento) è un accorgimento per rendere il dictionary attack più difficoltoso. Il salt è una stringa casuale che viene concatenata con le password, prima che queste vengano passate alla one-way function. Quindi, sia il valore del salt che i risultati della one-way function sono conservati nel database del computer host. Se il numero di possibili valori del salt è abbastanza grande, in pratica viene eliminata la possibilità di un dictionary attack contro le parole più comuni, perché Mallory dovrebbe generare il one-way hash per ogni possibile valore di hash. Il vero problema è assicurarsi che Mallory debba effettuare un tentativo di cifratura per ogni password del suo dizionario ogni volta che prova a violare la password di un'altra persona, piuttosto che poter fare un unico massiccio calcolo (fuori linea) per tutte le possibili password. Naturalmente la situazione migliora al crescere della dimensione del salt. La maggior parte dei sistemi UNIX, ad esempio, usa solo 12 bit di salt. A riguardo, Daniel Klein ha sviluppato un programma per indovinare le password che spesso, entro una settimana, riesce a

violare il 40% delle password di un dato host. Feldmeier e Karn, invece, hanno compilato una lista di circa 732.000 password comuni concatenate con tutti i 4096 possibili valori del salt, e stimano di poter violare circa il 30% di password su qualunque host. Il salt non è miracoloso; aumentandone il numero di bit non si risolve tutto. Il salt protegge solo da dictionary attack generici contro un file di password, non da un attacco contro una singola password. Protegge le persone che usano la stessa password su macchine diverse, ma non migliora le password scelte male.

Persino con il salt, il primo protocollo ha dei seri problemi di sicurezza. Quando Alice invia la propria password al computer host, chiunque abbia accesso al suo canale di comunicazione, può leggerla. Eve potrebbe essere da qualche parte lungo il percorso, pronta ad ascoltare la sequenza di login. Se Eve ha accesso alla memoria del processore del computer host, può vedere la password prima che sia passa alla one-way function. La crittografia a chiave pubblica può risolvere questo problema. Il computer host conserva un file con la chiave pubblica di ogni utente; ogni utente custodisce la propria chiave privata. Ecco un semplice esempio di protocollo. Quando si effettua il login il protocollo procede così:

1. Il computer host invia ad Alice una stringa casuale.

2. Alice cifra la stringa con la propria chiave privata e la rimanda al computer host, insieme al proprio nome.

3. Il computer host cerca la chiave pubblica di Alice nel proprio database e la usa per decifrare il messaggio.

4. Se la stringa decifrata è uguale a quella che il computer host ha inviato ad Alice, il computer host permette ad Alice di accedere al sistema.

La cosa più importante è che Alice non invia mai la sua chiave privata sulla linea di trasmissione; il risultato è che Eve, in questo protocollo, non può fare nulla. La chiave privata è lunga e difficile da ricordare e, di solito, è usata direttamente dal software (o hardware) di comunicazione dell'utente. Ciò richiede un terminale intelligente di cui Alice si possa fidare, ma, perlomeno, il canale e il computer host non devono necessariamente essere sicuri. Resta il fatto che, cifrare stringhe arbitrarie, soprattutto quando sono inviate da una terza parte di cui non ci si fida, è una cosa sciocca e può esporre ad alcuni tipi d'attacco. I protocolli di autenticazione sicuri hanno la seguente, più complessa, forma:

1. Alice esegue un calcolo basato su alcuni numeri casuali e sulla propria chiave privata ed invia il risultato al computer host.

2. Il computer host invia ad Alice un diverso numero casuale.

3. Alice esegue alcuni calcoli basati sui numeri casuali (sia quelli che ha generato che quello che ha ricevuto) e sulla propria chiave privata ed invia il risultato al computer host.

4. Il computer host esegue alcuni calcoli sui vari numeri ricevuti da Alice e sulla chiave pubblica di Alice per verificare se lei (Alice) conosce la sua (di Alice) chiave privata.

5. Se è così, la sua identità è verificata.

Se Alice non si fida del computer host più di quanto questo si fidi di lei, allora Alice pretenderà che il computer host provi la propria identità alla stessa maniera.

1.2.11 Bit commitment

La situazione è questa: Alice vuole fare una previsione (ad esempio un bit o una serie di bit), ma non la vuole rivelare fino a quando non si è avverata; Bob, d'altra parte, vuole assicurarsi che Alice non possa cambiare idea dopo essersi impegnata. Il protocollo (definito di bit commitment) che usa quindi la crittografia simmetrica può essere indicato di seguito:

1. Bob genera una sequenza di bit casuali, R, e la invia ad Alice.

R → Alice

2. Alice crea un messaggio costituito da i bit che vuole impegnare, b, e dalla sequenza casuale di Bob. Lo cifra con una chiave casuale, K, e invia il risultato a Bob.

EK(R + b) → Bob

Questa è la parte del protocollo in cui Alice si impegna. Bob non può decifrare il messaggio e così non può conoscere i bit. Quando per Alice arriva il momento di rivelare i bit, il protocollo continua:

3. Alice invia a Bob la chiave.

K → Bob

4. Bob decifra il messaggio per svelare i bit e controlla la sua sequenza casuale per verificare la validità dei bit.

DK[EK(R + b)] = R + b

Se il messaggio non contenesse la sequenza casuale di Bob, Alice potrebbe decifrare il messaggio inviato a Bob con diverse chiavi, fino a trovare quella che le dà il bit desiderato. Siccome un bit può assumere due soli valori, Alice troverà una chiave adatta dopo pochi tentativi. La sequenza di bit casuali previene questo attacco, perché Alice è costretta a produrre un messaggio con il suo bit invertito, ma con tutti quelli di Bob immutati. Se l'algoritmo di cifratura è buono, le sue possibilità di riuscita sono trascurabili.

Una variazione invece del protocollo di bit commitment che usa i generatori di sequenze pseudo-casuali può essere:

1. Bob genera una sequenza di bit casuali, RB, e la invia ad Alice.

2. Alice genera un seme casuale per un generatore di sequenze pseudo-casuali; quindi, per ogni bit della sequenza di Bob, invia a Bob l'output del generatore, se il bit di Bob è 0, oppure lo XOR dell'output del generatore e del proprio bit, se il bit di Bob è 1. Quando per Alice arriva il momento di rivelare i bit, il protocollo continua:

3. Alice invia a Bob il proprio seme.

4. Bob completa il passo 2 per confermare che Alice ha agito correttamente.

Una ulteriore variazione del protocollo mediante hashing può invece essere

1. Alice genera due stringhe di bit random, R1 e R2.

2. Alice crea un messaggio costituito dalle due stringhe e dai bit che vuole impegnare, b; di tale messaggio ne fa la hash.

H(R1 + R2 + b) = y

3. Alice invia a Bob sia y che la stringa R1.

{y, R1} → Bob

Questa è la parte del protocollo in cui Alice si impegna. Di seguito c'è la fase di verifica:

4. Alice invia a Bob il messaggio costituito dalle due stringhe e dai bit che vuole impegnare (a Bob mancava solo R2).

(R1 + R2 + b) → Bob

5. Bob ne fa la hash per verificare se il risultato ottenuto, y', coincide con quello ricevuto al passo 2, y.

H(R1 + R2 + b) = y'

1.2.12 Prove a conoscenza nulla

Purtroppo, di solito, l'unico modo di dimostrare di conoscere una certa cosa, consiste nel dirla; a quel punto, però, anche gli altri la sanno. Usando le one-way function, invece, Peggy può fornire una zero-knowledge proof. Questo protocollo dimostra a Victor che Peggy ha una certa informazione, ma non concede a Victor la possibilità di conoscere tale informazione. Questi protocolli sono di tipo interattivo. Victor pone a Peggy una serie di domande: se Peggy conosce il segreto, può rispondere correttamente a tutte le domande; se non lo conosce ha una certa probabilità di indovinare. Dopo un certo numero di risposte corrette, Victor può concludere che le prove di Peggy sono valide. Naturalmente i due possono accordarsi sulle domande, per cui una eventuale registrazione non può essere usata per convincere una terza parte della validità della prova. Supponiamo che Peggy conosca una certa informazione, e che tale informazione sia la soluzione di un problema difficile. Il protocollo zero-knowledge di base è composto da vari passi:

1. Peggy usa la propria informazione e un numero casuale per trasformare il problema difficile in un altro problema difficile, isomorfico rispetto a quello originale. Quindi usa la propria informazione e il numero casuale per risolvere questa nuova istanza del problema difficile.

2. Peggy mette da parte la soluzione della nuova istanza, usando uno schema di tipo bit commitment.

3. Peggy rivela a Victor la nuova istanza. Victor non può usare questo nuovo problema per ottenere informazioni riguardanti l'istanza originale o la sua soluzione.

4. Victor chiede a Peggy di dimostrargli che la vecchia e la nuova istanza sono isomorfe, oppure di rivelare la soluzione messa da parte al punto 2 e di dimostrare che è soluzione della nuova istanza.

5. Peggy fa quanto richiesto.

6. Peggy e Victor ripetono i passi da 1 a 5 per *n* volte.

Come anticipato, una eventuale registrazione del protocollo non può essere usata come prova, perché Peggy e Victor possono mettersi d'accordo. L'aspetto matematico che sta dietro a questo meccanismo è molto complicato. I problemi e la trasformazione che li lega devono essere scelti accuratamente, in modo che Victor non possa dedurre niente sulla soluzione del problema originale, anche dopo molte iterazioni del protocollo. Non tutti i problemi difficili possono essere usati per questo protocollo, ma, comunque, una buona parte di essi.

Isomorfismo tra grafi

Un grafo è una rete di linee che connettono un insieme di punti. Se due grafi differiscono solo per i nomi dati ai punti, sono detti isomorfi. Per un numero di punti grande, determinare se due grafi sono isomorfi può richiedere centinaia di anni; infatti, il problema è di tipo NP-completo. Supponiamo che Peggy conosca l'isomorfismo tra due grafi, *G1* e *G2*. Il seguente protocollo convince Victor del fatto che Peggy conosce una certa informazione:

1. Peggy permuta casualmente *G1* per ottenere un nuovo grafo, *H*, isomorfo rispetto a *G1*. Siccome Peggy conosce l'isomorfismo tra *H* e *G1*, conosce anche quello tra *H* e *G2*. Per chiunque altro, trovare un isomorfismo tra *H* e *G1* o tra *H* e *G2*, è difficile come trovarne uno tra *G1* e *G2*.

2. Peggy invia *H* a Victor.

3. Victor chiede ad Alice di:
   - dimostrare che H e G1 sono isomorfi, oppure,
   - dimostrare che H e G2 sono isomorfi.

4. Peggy fa quanto chiesto:
   - dimostra che *H* e *G1* sono isomorfi, senza dimostrare che anche *H* e *G2* lo sono, oppure,
   - dimostra che *H* e *G2* sono isomorfi, senza dimostrare che anche *H* e *G1* lo sono.

5. Peggy e Victor ripetono i passi da 1 a 4 *n* volte.

Se Peggy non conosce un isomorfismo tra *G1* e *G2*, non può creare un grafo *H* isomorfo ad entrambi. Quello che può fare è creare un grafo isomorfo rispetto solo a *G1* o *G2*. In questo modo ha il 50% di probabilità di indovinare quale prova le chiederà Victor al passo 3. Questo protocollo non dà a Victor nessuna informazione utile su come ottenere un isomorfismo tra *G1* e *G2*, perché, per ogni ripetizione del protocollo, Alice genera un nuovo grafo *H*.

1.3 Individuazione di procedure di calibrazione e diagnostica di tutti i misuratori afferenti alla rete di misura

In fase di progettazione si è pensato ad implementare delle procedure che permettono una *Self-identificati*on e *Self-configuration* per i singoli dispositivi – *Smart Meters* (SM)- connessi alla rete attraverso gli *Smart Concentrators (SC)*. È dunque opportuno precisare che, a differenza dello standard CAN, il sistema sviluppato implementa due ulteriori distinte fasi di comunicazione tra i dispositivi: il *Devices Setting State (DSS)* e lo Measurement *Data Exchange State (MDES)*. Per mantenere l'intrinseca robustezza del protocollo ed essere allo stesso tempo essere in grado di indirizzare correttamente tutti i dispositivi presenti nel sistema, il ruolo dello *SC* è anche quello di fornire indirizzi ai singoli *SMs* che in fase di start-up ne fanno richiesta (*DSS*). L'indirizzamento è basato sul campo *identifier* (11 bit) presente nel protocollo CAN cosicché è possibile connettere ad un singolo *SC* circa *2000 SMs*. Per quanto concerne il *MDES*, lo *SC* interroga ciclicamente (ad intervalli di 3 secondi) ogni *SM* attraverso un *Remote Frame Message* e aspetta la risposta prima di passare allo stato successivo (secondo una tabella di attivazione risultato del *DSS*).

1.4 Individuazione di procedure di intercambiabilità nella rete di controllo per misuratori installati

Le procedure adottate, insieme alla necessaria modifica del layer fisico, consentono la interconnessione di dispositivi realizzati da costruttori diversi anche operanti con differenti protocolli.

2 Realizzazione e caratterizzazione di un prototipo di rete di misuratori

Viene presentata una architettura hardware/software per la gestione dell'energia elettrica in tempo reale basata su sistemi a microcontrollori che forniscono funzionalità di *SM* per monitorare in maniera continua i carichi ad essi collegati e che comunicano ad una unità centrale - *Smart Concentrator* (*SC*) - i dati elaborati tramite CAN bus. Il compito principale di ogni

SC è recuperare temporaneamente i dati di misura da ogni singolo *SM* e renderli disponibili attraverso protocolli ed infrastrutture dell'*Internet of Things*. Attraverso un web server ed un database MySQL, siamo poi in grado di memorizzare tutte le informazioni prelevate dai singoli *SC* e fornire, ad esempio, agli utenti finali[2] il loro consumo effettivo utilizzando un semplice Web browser. Per rendere l'intero sistema sicuro e quindi prevenire attacchi esterni, prima delle diverse memorizzazioni lungo tutta la catena dei dispositivi coinvolti nello Smart Metering è stato implementato un algoritmo di crittografia basato su AES-128. Per implementare il sistema di misura in tempo reale rispetto alla grandezze di interesse della rete elettrica e rendere disponibili all'utente finale tali informazioni, l'architettura della Smart Meters Network (Figura 8) include:

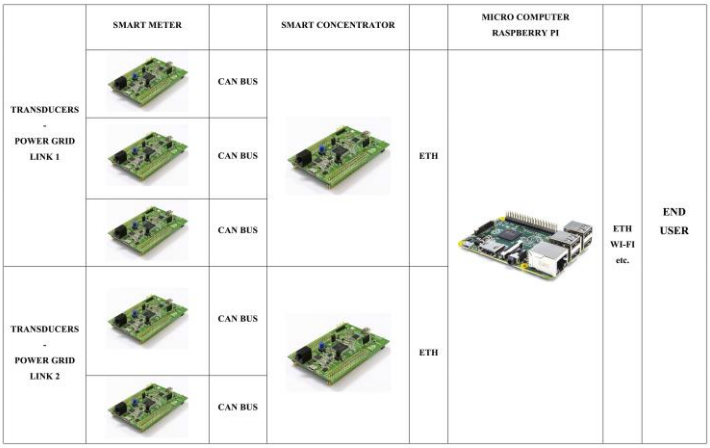


Figura 8: Connessioni Hardware

a) **n SMs** collegati ai diversi carichi (attraverso appositi trasduttori) che acquisiscono continuamente tensione e corrente misurandone Potenza (Attiva e Reattiva), Energia e diversi indici di *PQ*;

b) ***nodo aggregatore*** con funzionalità di *Smart Concentrator* (*SC*) collegato attraverso CAN bus ai singoli *SM* in grado di memorizzare su memory flash interna tutti i dati ricevuti e di metterli a disposizione per richieste HTTP verso l'esterno;

c) ***nodo interfaccia*** - *Micro Computer (MC)* - con funzionalità di memorizzazione su Database MySQL e diffusione verso l'utente finale (Web Server+applicazione Web-based con profilazione utente).

[2] Vi è la possibilità di impostare tre livelli di accesso al sistema: 1) utente domestico, 2) amministratore locale e 3) amministratore centrale elettrica con diverse funzionalità e privilegi.

Lo SM realizzato è costituito/implementato attraverso un microcontrollore *ARM Cortex M4 STM32F407* (Figura 9) in grado di proporre un vero e proprio Wattmetro composto da uno stadio di trasduzione (tensione e corrente) ed stadio di elaborazione (misura e visualizzazione). In particolare, è stato realizzato e testato sperimentalmente un prototipo di misuratore di energia elettrica in grado di effettuare le misurazioni con le principali metriche riportate nella letteratura scientifica internazionale, sia in ambito domestico sia industriale[3].

Figura 9: Prototipo dello SM - STM32F407

Per applicare utilmente le tecniche definite in questa fase è stato necessario progettare e realizzare specifici i trasduttori di corrente e tensione con tecnologia più idonea per la misura di potenza ed il rilievo dei parametri che determinano la qualità dell'alimentazione.

A tale scopo sono state quindi considerate alcune specifiche di progetto, quali la corrente massima misurata (che dovrà includere possibili elevati spike) la larghezza di banda (che dovrà garantire misure a frequenze dell'ordine di qualche kHz), unitamente ad un target di basso costo presunto del dispositivo complessivo. Il microcontrollore ha dunque il compito di acquisire i segnali di tensione e corrente, opportunamente condizionati dallo stadio di trasduzione, al fine di calcolare:

Valore efficace di tensione RMS_V	$RMS_V = \sqrt{\frac{1}{N}\sum_{k=1}^{N} v_k^{\,2}}$
Valore efficace di corrente RMS_I	$RMS_I = \sqrt{\frac{1}{N}\sum_{k=1}^{N} i_k^{\,2}}$

Potenza reattiva Q_{IEEE}	$Q = Q_{ieee} = \frac{1}{N}\left(\sum_{k=1}^{N} v_k \cdot \sum i_k\right)$
Potenza reattiva Q_{fryze}	$S_r = RMS_V \cdot RMS_I$ $Q_{fryze} = \sqrt{S_r^2 - P_A^2}$

[3] In maniera contestuale alle misure per la tariffazione saranno analizzati i principali indici di power quality, in maniera tale da poter associare alla quantità di energia tariffata anche un indice di livello qualitativo di fornitura che eventualmente può essere oggetto di variazione tariffaria concordata contrattualmente (Custom Power)

Potenza Attiva *P*	$P = \frac{1}{N}\sum_{k=1}^{N} v_k \cdot i_k$
Potenza reattiva Q_{ENEL}	$Q_{ENEL} = \frac{1}{N} \cdot \sum_{k=1}^{N} v_k \cdot i_{k+\frac{N}{4}}$
Distorsione armonica totale della tensione THD_V	$THD_V = \sqrt{\frac{\sum_{h=2}^{M} V_h^2}{V_1^2}}$

Potenza reattiva $Q_{budeanu}$	$Q_{budeanu} = \frac{1}{2} \cdot \sum_{h=1}^{M} V_h \cdot I_h \cdot \sin\varphi_h$
Pawer factor *P*	$PF = \frac{P}{A}$
Distorsione armonica totale della corrente THD_I	$THD_I = \sqrt{\frac{\sum_{h=2}^{M} I_h^2}{I_1^2}}$

Per la trasduzione dei segnali elettrici di tensione e corrente si è pensato ad una sezione di trasduzione in grado di alimentarsi dal segnale in ingresso. I trasduttori sono stati così progettati in modo che con un segnale sinusoidale di ingresso pari al valore di progetto (460 RMS_V per quello di tensione e 30 RMS_I per quello di corrente) si abbia per entrambi in uscita una tensione sinusoidale pari a 1.5 V (valore di picco) e una componente continua pari a 1.5V. Per la tensione si è utilizzato un valore di sovraccarico 2 per tenere conto di eventuali sovratensioni. La figura di seguito riporta una foto del prototipo realizzato.

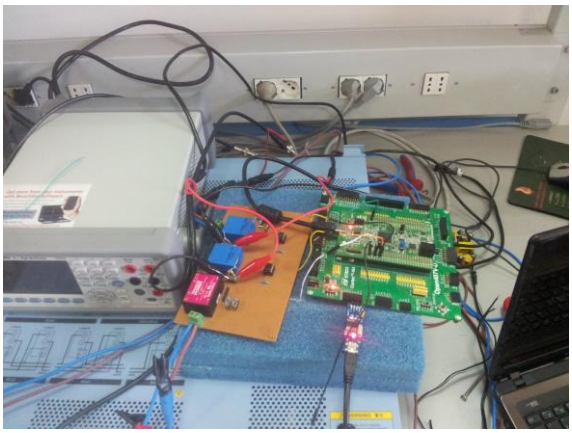


Figura 10: Prototipo di Smart Meter con sezione di Trasduzione

Lo Smart Meter realizzato è stato programmato per effettuare:

- acquisizione sincrona di due segnali (le uscite del trasduttore di corrente e di tensione);
- calcolo di frequenza, valori efficaci, potenza attiva, energia attiva, potenza apparente, energia apparente, fattore di potenza, potenza ed energia reattiva secondo le principali definizioni accreditate in letteratura, misura di THD_I e THD_V;
- mostrare i risultati e prendere comandi tramite un'interfaccia locale utente (DISPLAY LED);

- memorizzazione dei risultati di misura in locale (Memoria Flash locale) per un determinato intervallo di tempo (gestione attraverso buffer circolare);
- trasferimento dei dati acquisiti attraverso interfaccia CAN verso il nodo concentratore a richiesta e in modalità programmata temporizzata.

Anche lo Smart Concentrator (*SC*) è costituito da un sistema microcontrollore *ARM Cortex M4 STM32F407* dotato di una memoria locale per l'immagazzinamento temporaneo dei dati ed una unità di comunicazione (sono state implementate in questo dispositivo due tipi di interfaccia: una rivolta agli Smart Meters con protocollo CAN ed una rivolta al *Nodo Interfaccia* di tipo Ethernet), per la interazione con i dispositivi di campo e la pubblicazione in rete (trasmissione dei dati a livello superiore) degli stessi.

Il consumatore come il gestore locale della rete elettrica ha, quindi, la possibilità di monitorare in locale (attraverso Display a Led) piuttosto che da remoto (attraverso un Server Web) i consumi relativi alle utenze di proprio interesse. Lo SC, una volta recuperati i dati - comunicazione attraverso CAN BUS - dai singoli SM, ne verifica la correttezza (sintattica), l'immagazzina per renderli disponibili ad un'analisi di primo livello e li trasferisce al *nodo interfaccia* quando da questi ne viene fatta richiesta. L'attività dello SC si divide in *Device Setting State* durante la quale ogni singolo SM che si collega alla sua sottorete chiede un indirizzo attraverso il quale verrà poi interrogato ed in *Measurement Data Exchange State* attraverso cui viene effettuata (in base alla tabella di attivazione prodotta nello stato precedente) una interrogazione ciclica (arbitraggio del CAN BUS in Sequential Polling) su tutti gli SM attivi. Una volta recuperati i dati, lo SC li mette a disposizione rispondendo a richieste di tipo http, fungendo da Server Web in modo che le richieste possano essere effettuate sia da Web Browser sia dal nodo interfaccia attraverso un'applicazione ad hoc in linguaggio dinamico php-javascript in grado di scaricare le Memorie Flash dei singoli SC all'interno di un Database MySql dedicato.

Il microcomputer è l'ultimo tassello dell'aggregazione di informazioni ed è in grado di rendere accessibili, tramite interfaccia user-friendly, la totalità dei dati ricevuti dall'intera catena di misura *SMs + SCs*. Su di esso vengono avviati un *web-server Apache* ed un'applicazione web-based (php, javascript e html) per gestire le richieste HTTP dell'utente finale, nonché un *Database MySQL* per la memorizzazione di tutti i dati recuperati dai diversi *SCs*. Dopo la fase di autenticazione, il primo passo è dunque scaricare i dati dal singolo SC accessibile tramite il suo indirizzo IP (gestito secondo la rete locale nella quale è inserito il nodo) come mostrato in Figura 11 per trasferire le misure presenti nella memoria flash dei singoli dispositivi in un DB MySQL *crittografato*.

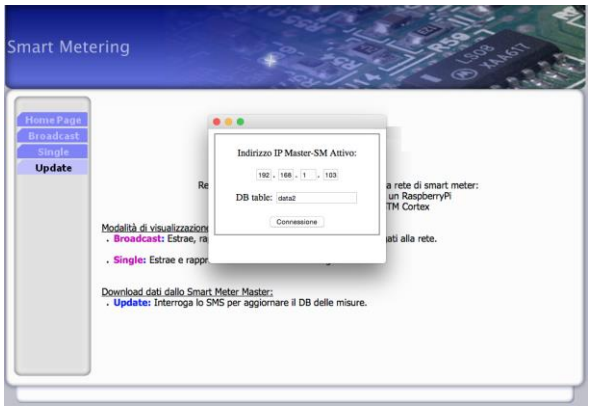


Figura 11: Interfaccia per il trasferimento dei dati dal singolo SC al Nodo Aggregatore

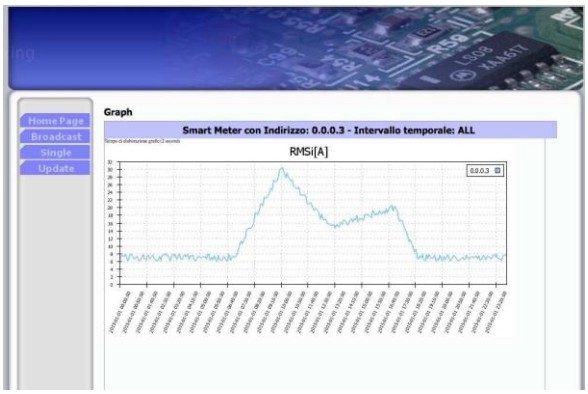


Figura 12: Grafico riferito al RMS_I per lo SM 0.0.0.3

Una volta acquisiti i dati di interesse, un esempio di possibile visualizzazione fornita dall'applicazione web relativamente ai consumi monitorati è di seguito mostrata in Figura 12 con la rappresentazione di un carico/SM *0.0.0.3*.

2.1.1 Breve panoramica sui microprocessori ARM

L'architettura ARM (precedentemente Advanced RISC Machine, prima ancora Acorn RISC Machine) indica una famiglia di microprocessori RISC a 32-bit sviluppata da ARM Holdings e utilizzata in una moltitudine di sistemi embedded. Grazie alle sue caratteristiche di basso consumo (rapportato alle prestazioni) l'architettura ARM domina il settore dei dispositivi mobili dove il risparmio energetico delle batterie è fondamentale. Attualmente la famiglia ARM copre il 75% del mercato mondiale dei processori a 32 bit per applicazioni embedded, ed è una delle più diffuse architetture a 32 bit del mondo. I processori ARM vengono utilizzati in PDA, cellulari, tablet, lettori multimediali, videogiochi portatili e periferiche per computer (come router, hard disk di rete ecc.).

Microcontrollori basati su ARM Cortex-M

Il Microcontrollori della famiglia ARM Cortex-M Series sono processori embedded ottimizzati per applicazioni a basso costo. Questi processori supportano il set di istruzioni Thumb-2. L' ARM Cortex-M4 è l'ultimo processore embedded della famiglia ARM Cortex-M, nato come evoluzione del core ARM Cortex™-M3. Il core M4 offre prestazioni avanzate di controllo digitale e quindi ideale per applicazioni DSC, ovvero applicazioni di controllo ed elaborazione del segnale. Le sue caratteristiche sono basso assorbimento ed alta efficienza, è particolarmente adatto alle applicazioni in real time ed è stato progettato per rispondere alle richieste di soluzioni flessibili specificamente mirate a campi come il motor control, auto motive, gestione energetica, sistemi embedded audio e dell'automazione industriale. Il Cortex-M4F è un processore con le stesse

capacità del processore Cortex-M4, e include le funzionalità di aritmetica in virgola mobile (FPU). Gli ARM Cortex-M3 sono molto simili nelle caratteristiche agli M4, ma sono indirizzati a quelle applicazioni embedded che richiedono una risposta rapida agli interrupt, come sistemi di controllo auto motive e industriali. L'ARM Cortex-M1 FPGA è stato progettato per applicazioni embedded che richiedono un piccolo processore integrato in un FPGA. L'ARM Cortex-M0 e M0 + sono processori ad alta efficienza energetica destinati a microcontrollori e applicazioni embedded che richiedono un consumo di potenza e dimensioni del processore ottimizzate. Gli SM e il SC sono stati implementati attraverso il Cortex-M4 che è un processore a 32 bit ad alto rendimento. Offre notevoli vantaggi per gli sviluppatori, tra cui:

- elevatisime prestazioni di elaborazione combinate con una veloce gestione degli interrupt.
- sistema di debug migliorato.
- processor core e memoria ad elevate performance
- bassissimo consumo di potenza nelle diverse modalità di standby e sleep mode

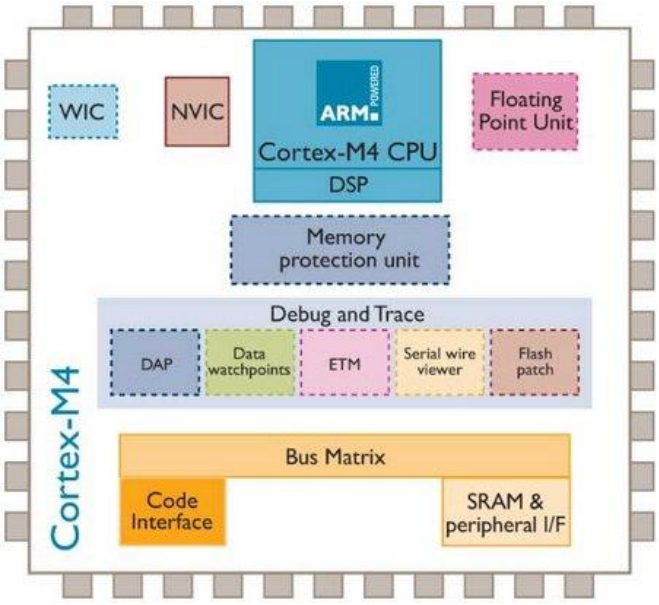


Figura 13: Cortex-M4 implementation

Il processore Cortex-M4 si basa su di un core ad alte prestazioni, con una architettura Harvard pipeline a tre fasi, che lo rende ideale per le applicazioni di tipo embedded. Il processore assicura un'eccezionale efficienza energetica attraverso un set di istruzioni efficiente ed ottimizzato per facilitare la progettazione di dispositivi a basso costo. Implementa i componenti del sistema strettamente accoppiati in modo tale da ridurre significativamente le dimensioni del processore migliorando sensibilmente la gestione delle interrupt e le funzionalità di debug di sistema così come una versione del set di istruzioni Thumb ® basato sulla tecnologia Thumb-2,

garantendo un'elevata densità di codice. Il set di Istruzioni del Cortex-M4 fornisce le prestazioni eccezionali che ci si aspetta da una moderna architettura a 32 bit, con elevata densità di codice. Il processore Cortex-M4 si integra strettamente con l'unità standard Nested Vectored Interrupt Controller (NVIC), per offrire elevate prestazioni nella gestione delle interrupt. L'NVIC include un interrupt non mascherabile (NMI) che può fornire fino a 256 di priorità di interrupt. La stretta integrazione del core e dell'unità NVIC fornisce una rapida esecuzione delle routine di servizio di interrupt (ISR), riducendo drasticamente la latenza degli interrupt. Questo risultato è ottenuto attraverso la sovrapposizione dei registri hardware, e la possibilità di sospendere le operazioni di multiple di caricamento e memorizzazione. Infine fornisce interfacce multiple che gestisce utilizzando AMBA (Advanced Microcontroller Bus Architecture) garantendo così un elevata velocità e bassa latenza sugli accessi alla memoria.

Debug Integrato e Periferiche interne

Il processore Cortex-M4 implementa un debugger integrato, che fornisce visibilità del sistema del processore e della memoria sia attraverso una tradizionale porta JTAG o una porta SWD (Serial Wire Debug) che è l'ideale per i microcontrollori e altri dispositivi di piccole dimensioni. Include inoltre molte periferiche come:

- *Nested Vectored Interrupt Controller (NVIC)* è un controller di interrupt integrato che supporta l'elaborazione a bassa latenza degli interrupt.
- *System Control Block (SCB)* fornisce informazioni sul sistema di attuazione e controllo del sistema. Questo include la configurazione, il controllo e il reporting delle eccezioni di sistema.
- *SysTick:* è il timer di sistema, è un count down timer a 24-bit. Può essere utilizzato come RTOS (Real Time Operating System) o come un semplice contatore.
- *Unità di protezione della memoria (MPU)* migliora l'affidabilità del sistema, definendo le caratteristiche di memoria per le regioni di memoria differenti.
- *Floating-point Unit: f*ornisce la possibilità di effettuare operazioni conformi allo standard IEEE 754 per operazioni a singola precisione, a 32 bit, e valori in virgola mobile

Pipeline, eccezioni e interrupt

La CPU del Cortex è in grado di eseguire più istruzioni in un unico ciclo. In entrambi le architetture dell'ARM7 e dell'ARM9, questo è realizzato con una pipeline a tre stadi ovvero mentre

esegue un'istruzione, la prossima viene decodificata e la successiva viene prelevata dalla memoria. Supporta anche la branch prediction per minimizzare il numero di flush della pipeline.

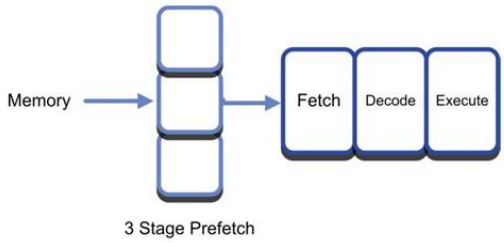


Figura 14: schema logico pipe line

Oltre alla pipeline, il processore Cortex-M4 attraverso la NVIC supporta gli interrupt e le eccezioni del sistema.

Interruzioni

Ogni interrupt può trovarsi in uno dei seguenti stati:

- **Inattivo**: L'eccezione non è ne attiva ne in attesa di essere servita.
- **In Attesa**: L'eccezione è in attesa di essere servita dal processore. Una richiesta di interrupt da una periferica o dal software può cambiare lo stato di attesa corrispondente all'interrupt.
- **Attivo**: l'eccezione è gestita da parte del processore, ma non è stata ancora completata[4].
- **Attivi e in sospeso**: L'eccezione è gestita dal processore e vi è un'altra eccezione in sospeso.

Tipi di eccezioni

Il microcontrollori basati su ARM CORTEX M4 possono gestire le seguenti tipologie di interruzione

- **Reset**: Il reset è un'interruzione particolare che ha il livello massimo di priorità, se un'interruzione di reset viene richiamata allora il processore si interrompe indipendentemente dall'operazione che sta svolgendo e il sistema viene riavviato.
- **NMI (Non maskable interrupt)**

  Un interrupt non mascherabile (NMI) è un tipo particolare di interrupt che non può essere ignorato dalla CPU questa eccezione ha una priorità diversa dal reset. È sempre attiva e ha una priorità fissa di -2. NMI , può essere utilizzato da una periferica per segnalare un malfunzionamento.

[4] Un gestore di eccezione può interrompere l'esecuzione di un altro gestore di eccezioni. In questo caso entrambe le eccezioni sono nello stato attivo

- **HardFault**

  Un HardFault è un'eccezione che si verifica a causa di un errore durante l'elaborazione delle eccezioni. Le eccezioni di tipo HardFaults hanno una priorità fissa di -1, nel senso che hanno una priorità più alta rispetto a qualsiasi eccezione con priorità configurabile.

- **MemManage**

  Un guasto MemManage è un'eccezione che si verifica a causa di un errore di protezione della memoria.

- **BusFault**

  Un BusFault è un'eccezione che si verifica quando un programma tenta di accedere ad una posizione di memoria alla quale non gli è permesso accedere, oppure quando tenta di accedervi in una maniera che non gli è concessa. Questo potrebbe essere rilevato come un errore sul bus nel sistema di memoria.

- **UsageFault**

  Un UsageFault è un'eccezione che si verifica a causa di un errore che riguarda l'esecuzione delle istruzioni. Come ad esempio un'istruzione non definita, un accesso non allineato in memoria o una divisione per zero.

- **SysTick**

  Un'eccezione SysTick è un'eccezione generata da un Timer (quando quest'ultimo raggiunge lo zero).

- **Interrupt (IRQ)**

  Un allarme, o IRQ, è un'eccezione segnalata da una periferica, o generata da una richiesta del software. Tutti gli allarmi sono asincroni rispetto all'esecuzione delle istruzioni. Questo tipo di interrupt è spesso utilizzato dalle periferiche per comunicare con il processore.

Gestori delle eccezioni

Il processore gestisce le eccezioni utilizzando:

- **ISR** (Interrupt routine di servizio), tutti gli interrupt di tipo IRQ sono eccezioni gestite dall' ISR.
- **Fault handlers** (Gestori d'errore) le interruzioni di tipo HardFault, MemManage fault, UsageFault, e BusFault sono eccezioni trattate dai gestori di errore.
- **System handlers** (gestori del sistema) NMI, PendSV, SVCall SysTick, sono tutte eccezioni di sistema che vengono gestite da gestori del sistema

Risparmio energetico

I processori ARM CORTEX i più utilizzati nel settore dello sviluppo di dispositivi mobili proprio per la loro capacità di fornire elevate prestazioni e consumi energetici ridotti. I processori cortex M4 implementano due modalità per ridurre il consumo energetico del processore:

- **Sleep Mode** (modalità di sospensione) si ferma il clock del processore
- **Deep Sleep Mode** (modalità di sonno profondo) si ferma l'orologio del sistema, si spegne il PLL e la memoria flash.

Possono essere implementate entrambe le modalità per fornire vari livelli di risparmio energetico.
Il core Cortex può essere messo in modalità 'sleep' dall'esecuzione dell'istruzione Wait For Interrupt (WFI) o dal Wait For Event (WFE). In caso dell'istruzione WFI, il core Cortex si riattiverà e servirà l'interrupt. Una volta servito l'interrupt ci sono due possibilità, la prima è che la CPU ritorni all'esecuzione del codice sorgente, oppure settando nel registro di sistema di controllo il bit SLEEPON EXIT, il core Cortex entrerà automaticamente nella modalità 'sleep' dopo che ha servito l'interrupt. Ciò ci permette di risparmiare potenza. L'istruzione WFE permette al Cortex di riprendere l'esecuzione dal punto in cui era stato messo in modalità sleep cioè all'avvento di un interrupt la CPU non salta alla service routine. Il 'risveglio' può essere causato da un interrupt di una periferica non abilitato nell'NVIC. Le istruzioni WFI e WFE non possono essere programmate in linguaggio C, ma il Thumb-2 instruction set presenta delle macro che possono essere inserite all'interno del codice C per richiamare tali istruzioni.

Floating-point unit (FPU)

L' FPU supporta pienamente tutte le operazioni in singola precisione. Fornisce inoltre le conversioni tra formati di dati a virgola fissa e virgola mobile. la FPU fornisce funzionalità di calcolo in floating point che sono compatibili con lo standard ansi / ieee std 754-2008 Lo standard IEEE per il calcolo in virgola mobile (IEEE 754- 2008) è lo standard più diffuso nel campo del calcolo automatico. Questo standard definisce il formato per la rappresentazione dei numeri in virgola mobile , ed un set di operazioni effettuabili su questi. Specifica inoltre quattro metodi di arrotondamento e ne descrive cinque eccezioni.

Microcontrollori STM32

La famiglia STM32 di microcontrollori flash a 32 bit basati su processore ARM Cortex-M, comprende una serie di prodotti a 32 bit che abbinano prestazioni elevate, funzionalità in tempo reale, elaborazione di segnali digitali e funzionamento a bassa potenza e bassa tensione, pur mantenendo caratteristiche di totale integrabilità e facilità di sviluppo.

Serie STM32 F4

La serie STM32 F4 basata su ARM® Cortex™-M4 è un'estensione del portafoglio STM32, leader del settore, in grado di offrire prestazioni ancora più elevate. Questi MCU si avvalgono della tecnologia NVM a 90 nm di ST e di ART Accelerator di ST per raggiungere i massimi punteggi di benchmark del settore per microcontrollori basati su Cortex-M con 210 DMIPS e punteggio Coremark 469 eseguendo il benchmark dalla memoria flash a una frequenza di funzionamento di 168 MHz. Le istruzioni DSP e l'unità di calcolo a virgola mobile ampliano il numero di applicazioni nelle quali è possibile utilizzare questi MCU. La serie STM32 F4 è il risultato di una simbiosi perfetta delle capacità di controllo in tempo reale di un MCU e delle prestazioni di elaborazione dei segnali di un DSP e quindi arricchisce il portafoglio STM32 con una nuova classe di dispositivi: controllori per segnali digitali (DSC).

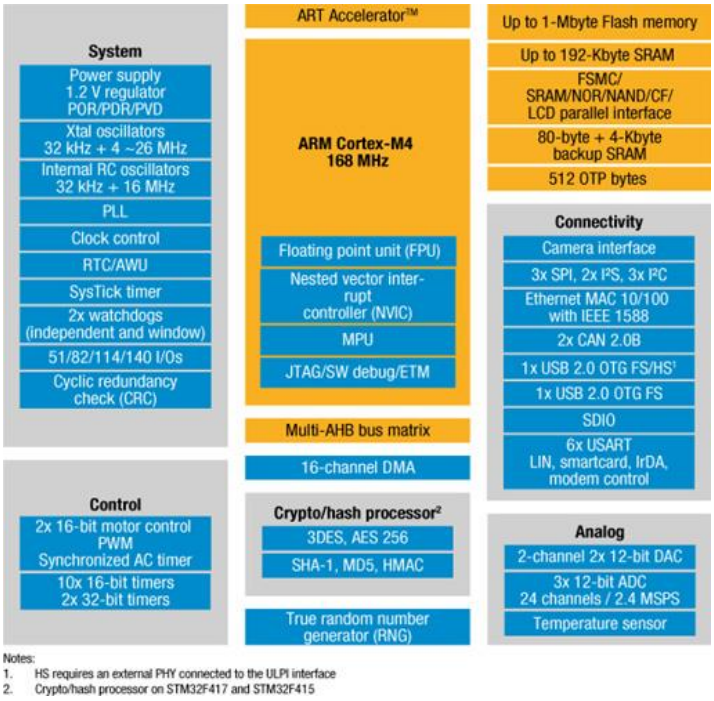


Figura 15: Schema a blocchi del microcontrollore ARM Cortex M4

Clock e avvio

Quando il microcontrollore viene avviato, un oscillatore interno (RC) a 16 MHz è selezionato di default come clock della CPU, questo oscillatore è costruito per offrire un'accuracy dell'1% nell'intero range di temperatura di funzionamento. Nelle nostre applicazioni possiamo utilizzare come sorgente del Clock sia l'oscillatore RC che un oscillatore esterno, che può avere una frequenza di funzionamento tra i 4Mhz ed i 26MHz. Se si seleziona un clock esterno quest'ultimo può essere monitorato, in maniera tale che se dovesse essere rilevato un malfunzionamento il

sistema passa automaticamente all'oscillatore interno (RC) generando un interrupt software quando quest'ultimo viene abilitato. La sorgente di clock va in ingresso ad un PLL che permette di portare la frequenza di funzionamento fino ad un massimo di 168 MHz. Diversi Prescaler consentono la configurazione di tre bus AHB (Advanced High Bus), dell' high speed APB2 (Advanced peripheral bus two) e del low speed APB1 (Advanced peripheral bus one). La massima frequenza dei tre bus AHB è 168 MHz, mentre la massima frequenza agli high speed APB è 84 MHz e ai low speed APB è 42 MHz. Tre differenti sorgenti di clock possono essere utilizzate per pilotare il clock di sistema (SYSCLK):

- HSI oscillator clock
- HSE oscillator clock
- Main PLL (PLL) clock

I dispositivi hanno le due seguenti fonti di clock secondarie:

- Low-speed internal RC (LSI RC) a 32 kHz che può guidare il watch dog, L'RTC può essere utilizzato come sveglia dalle modalità di stop e standby.
- Low-speed external crystal (LSE crystal) 32,768 kHz che può pilotare eventualmente l'orologio RTC (RTCCLK)

Per ottimizzare i consumi di energia ogni sorgente di clock può essere attivata o disattivata in modo indipendente quando non viene utilizzata.

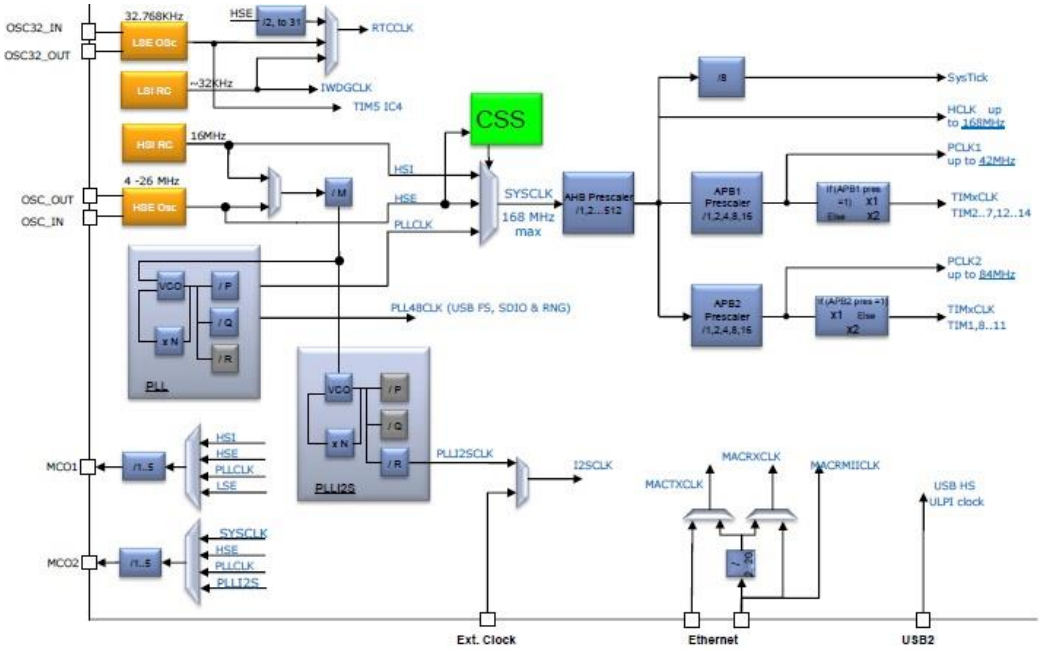


Figura 16: Schema del Clock del microcontrollore ARM Cortex M4

Il Clock controller fornisce un elevato grado di flessibilità nella scelta dell'oscillatore e della sua frequenza per avviare il core e le periferiche, allo stesso tempo garantisce la frequenza

appropriata per periferiche che richiedono un clock specifico come l'Ethernet, USB, OTG, FS, I2S, SDIO ed altre.

2.1.2 Micro Computer – Raspberry PI

Il micro computer è l'ultimo tassello dell'aggregazione di informazione ed è in grado di rendere accessibile attraverso interfaccia user-friendly l'insieme dei dati prelavati dall'intera catena di misura (SMs+SCs). Su di esso sono stati quindi implementati:

- Un Web Server Apache in grado di accettare richieste in HTTP;
- Un database opensource MySQL che memorizza i dati acquisiti dai singolo SC;
- Un'applicazione Web based (php, javascript e HTML) per gestire una interfaccia utente che consente di monitorare in tempo reale l'andamento della rete;

Nella fattispecie è stato utilizzata un'architettura Linux-based che rende flessibile e semplice l'integrazione dei dati/servizi. Il Raspberry Pi è un single-board computer (un calcolatore implementato su una sola scheda elettronica) sviluppato nel Regno Unito dalla Raspberry Pi Foundation. Il suo lancio al pubblico è avvenuto alla fine del mese di Febbraio 2012. L'idea di base è la realizzazione di un dispositivo economico, concepito per stimolare l'insegnamento di base dell'informatica e della programmazione nelle scuole. Il progetto ruota attorno a un System-on-a-chip (SoC) Broadcom BCM2835, che incorpora un processore ARM1176JZF-S a 700 MHz, una GPUVideoCore IV con 512 MB di memoria RAM. Il progetto non prevede nè hard disk nè un'unità a stato solido, affidandosi invece a una scheda SD per il boot e per la memoria non volatile. La scheda è stata progettata per ospitare sistemi operativi basati su un kernel Linux o RISC OS.

Figura 17: Raspberry Pi Model A

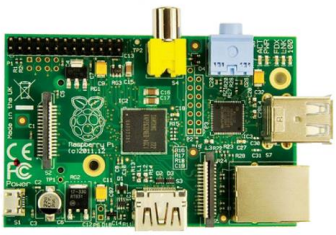


Figura 18: Raspberry Pi Model B

Software e Sistema Operativo

La Raspberry Pi Foundation ha messo a disposizione per il download un proof of concept di immagine che può essere caricata su SD card per produrre un sistema operativo preliminare. L'immagine si basa su Debian 6.0 (Squeeze), con un ambiente desktop LXDE e un browser Midori,

più vari strumenti di programmazione. L'immagine può anche girare sull'emulatore QEMU, permettendo di emulare Raspberry Pi su varie altre piattaforme. La Fondazione ha realizzato una release ottimizzata di Fedora, raccomandandola come sistema operativo. È disponibile anche una versione di Arch Linux. Esistono distribuzioni per l'utilizzo del Raspberry Pi come Media Center basate su XBMC: OpenELEC, XBian e RaspBMC. Il software di monitoraggio di rete Overlook Fing è stato portato su piattaforma Raspberry Pi rendendo possibile l'installazione di sentinelle di monitoraggio a basso costo in reti remote.Il software open source Aseba per la programmazione semplice ed efficiente di robot è disponibile su Raspberry PI. Utilizzando il Raspberry PI in unione con Aseba e il robot Thymio II è possibile creare a costi veramente contenuti un vero e proprio laboratorio didattico di Robotica. Il robot Thymio II è stato sviluppato nell'ambito del programma NCCR Robotics dalla collaborazione tra l'École Polytechnique Fédérale de Lausanne (EPFL) e l'École Cantonale d'Art de Lausanne (ECAL).

	Model A	**Model B**
Prezzo di offerta:	USD 25 (GBP16)	USD 35 (GBP 22)
SoC:	Broadcom BCM2835 (CPU + GPU + DSP + SDRAM)	
CPU:	700 MHz ARM1176JZF-S core (famiglia ARM11)	
GPU:	Broadcom VideoCore IV, OpenGL ES 2.0, 1080p30 H.264 high-profile decode	
Memory (SDRAM):	256 Megabytes (condivisa con la GPU)	256 o 512 Megabytes (condivisa con la GPU)
USB 2.0 ports:	1	2 (attraverso un hub USB integrato)
Output video:	Connettore RCA per il video composito, HDMI	
Output audio:	3,5 mm jack, HDMI	
Memoria:	SD / MMC / SDIO card slot	
Collegamenti di rete:	Nessuno	Ethernet 10/100 (RJ-45)
Periferiche di basso livello:	2x13 header pins for GPIO, SPI, I^2C, UART, +3,3 Volt, +5 Volt	
Real-time clock:	No clock or battery	
Corrente (potenza) assorbita:	300 mA, (1,5 W)	700 mA, (3,5 W)
Alimentazione:	5 V via MicroUSB o GPIO header	
Dimensioni:	85,60 mm × 53,98 mm (3.370 inch × 2.125 inch)	
Sistemi operativi supportati:	Debian GNU/Linux, Fedora, Arch Linux e Gentoo	
Sistemi operativi non supportati:	RISC OS (shared source)	

Tabella 19: Specifiche dei due modelli di Raspberry Pi

Formati supportati

Dato che il progetto ha come obiettivo la riduzione dei costi, la decodifica in hardware di alcuni formati multimediali non è supportata perchè richiede una specifica licenza. Il dispositivo può riprodurre in hardware formati liberi, quali H.264, mentre per riprodurre i formati MPEG-2 e VC-1 è possibile acquistare la relativa licenza abilitando l'hardware alla decodifica

2.2 Individuazione di indici di prestazione

Le caratteristiche dell'architettura della rete di misuratori e dei protocolli scelti/implementati sui prototipi SM, SC e MC hanno confermato le buone prestazioni dell'intero sistema. Si può evidenziare che:

- *l'incertezza di misura riscontrata, nelle diverse condizioni operative non è stata mai superiore allo 0.5%;*
- *resilienza ad errori di comunicazione*: la gestione degli errori di comunicazione è affidata alla robustezza dei due protocolli (CAN ed Ethernet) che nei diversi livelli introdotti garantiscono elevate prestazioni in termini di ritrasmissione in caso di perdita dei dati;
- *tempo di latenza sulla comunicazione*: i ritardi presenti nel sistema di comunicazione sono relativamente bassi oltre che garantiti dai protocolli standard utilizzati. Con riferimento a possibili criticità che si possono manifestare (data la possibile ampia estensione della rete) il sistema di immagazzinamento locale su memoria flash interna dei singoli dispositivi supera il problema della latenza rendendo comunque l'intero sistema robusto.
- *sicurezza del dato*: lungo tutta la catena di misura, così come per il database MySql, è stata sfruttata un'efficiente procedura di crittografia, secondo algoritmo AES-128, che in questo momento garantisce la comprensibilità delle informazioni solo agli utenti autorizzati.
- *minimo intervallo di lettura*: gli SM effettuano misurazioni in linea/tempo reale grazie ad un sistema efficiente di gestione dei convertitori AD e mettono a disposizione dei relativi SC i dati ogni 3 secondi. Per disporre, invece, dei dati all'interno del DB su MC, sono state implementate due scelte alternative ovvero scaricare gli SC ad intervalli regolari (impostato ogni 5 minuti, ma può variare in funzione dei SM collegati ad ogni SC) piuttosto che interrogare puntualmente in qualsiasi momento lo SC di interesse.

2.3 *Caratterizzazione metrologica in laboratorio.*

Al fine di caratterizzare la funzionalità del prototipo realizzato e la corretta implementazione degli algoritmi di misura è stata implementata una stazione di taratura come mostrato in Figura 20.

Figura 20: Stazione allestita per la caratterizzazione

Sono stati quindi utilizzati i seguenti strumenti:

- FLUKE 6105 (Caratteristiche: 1 kV e 21 A disponibile per ogni fase, controllo completo indipendente di tensione e corrente su ogni fase, fino a 100 armoniche e DC su uno o entrambi i canali di tensione e corrente, Potenza in Ingresso : Tensione da [100V - 240V] fino a ± 10% di fluttuazione, frequenza da [47-63] Hz);
- YOKOGAWA WT3000 (Caratteristiche: Accuracy di Potenza dello 0,02% di lettura + 0,049% di range, intervallo di frequenza DC e [0,1Hz-1MHz]);

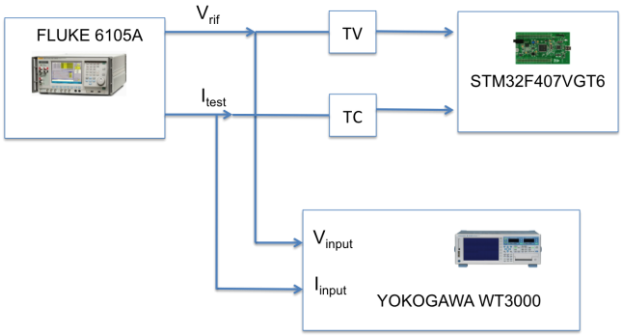


Figura 21: Connessione strumentazione per caratterizzazione SM

Mediante il sistema sopra descritto sono state effettuate le prove per la taratura del prototipo, generando forme d'onda di corrente e tensione, sinusoidali ed a frequenza nominale

relativamente a 10 ripetizioni per ogni misura e calcolato il valor medio e la deviazione standard come indicato nella tabella posta di seguito:

Tabella 22: Risultati di caratterizzazione in termini assoluti

Vset	RMSV [V]	µRMSV	Iset[A]	RMSI [AV]	µRMSI	cos(Φ)set	cos(Φ)	µcos(Φ)	fsset	fs	µfs	Pset	P	µP	Sset	S	µS
230,00	230,0	0,9	1,000	1,00	0,05	0,8000	0,800	0,003	49,000	49,00	0,03	184,0	184	22	230,0	230	28
230,00	230,0	1,2	5,000	5,00	0,12	0,8000	0,800	0,004	49,000	49,00	0,01	920,0	920	15	1150,0	1150	26
230,00	230,0	0,7	10,000	10,00	0,14	0,8000	0,800	0,003	49,000	49,00	0,03	1840,0	1840	21	2300,0	2300	26
230,00	230,0	1,6	10,000	10,00	0,07	0,9000	0,900	0,002	49,000	49,00	0,02	2070,0	2070	28	2300,0	2300	14
230,00	230,0	1,3	1,000	1,00	0,09	0,9000	0,900	0,002	49,000	49,00	0,03	207,0	207	12	230,0	230	25
230,00	230,0	1,0	5,000	5,00	0,15	0,9000	0,900	0,004	49,000	49,00	0,03	1035,0	1035	17	1150,0	1150	15
230,00	230,0	0,7	10,000	10,00	0,11	1,0000	1,000	0,005	49,000	49,00	0,02	2300,0	2300	18	2300,0	2300	13
230,00	230,0	0,7	10,000	10,00	0,15	0,8000	0,800	0,004	50,000	50,00	0,03	1840,0	1840	21	2300,0	2300	26
230,00	230,0	1,0	1,000	1,00	0,05	1,0000	1,000	0,003	49,000	49,00	0,03	230,0	230	22	230,0	230	11
230,00	230,0	0,9	5,000	5,00	0,05	1,0000	1,000	0,003	49,000	49,00	0,02	1150,0	1150	12	1150,0	1150	11
230,00	230,0	0,9	10,000	10,00	0,09	0,9000	0,900	0,004	50,000	50,00	0,02	2070,0	2070	17	2300,0	2300	17
230,00	230,0	1,2	10,000	10,00	0,10	1,0000	1,000	0,006	50,000	50,00	0,03	2300,0	2300	17	2300,0	2300	29
230,00	230,0	0,6	1,000	1,00	0,10	0,8000	0,800	0,002	50,000	50,00	0,02	184,0	184	21	230,0	230	18
230,00	230,0	1,1	5,000	5,00	0,06	0,8000	0,800	0,002	50,000	50,00	0,04	920,0	920	11	1150,0	1150	15
230,00	230,0	1,3	10,000	10,00	0,11	0,8000	0,800	0,005	51,000	51,00	0,03	1840,0	1840	16	2300,0	2300	24
230,00	230,0	1,6	10,000	10,00	0,09	0,9000	0,900	0,005	51,000	51,00	0,02	2070,0	2070	14	2300,0	2300	14
230,00	230,0	1,1	1,000	1,00	0,14	0,9000	0,900	0,003	50,000	50,00	0,03	207,0	207	16	230,0	230	16
230,00	230,0	1,1	5,000	5,00	0,07	0,9000	0,900	0,006	50,000	50,00	0,03	1035,0	1035	21	1150,0	1150	16
230,00	230,0	0,6	10,000	10,00	0,08	1,0000	1,000	0,003	51,000	51,00	0,02	2300,0	2300	14	2300,0	2300	22
230,00	230,0	1,6	20,000	20,00	0,07	0,8000	0,800	0,004	49,000	49,00	0,03	3680,0	3680	30	4600,0	4600	18
230,00	230,0	1,5	1,000	1,00	0,05	1,0000	1,000	0,004	50,000	50,00	0,03	230,0	230	19	230,0	230	24
230,00	230,0	0,7	5,000	5,00	0,06	1,0000	1,000	0,005	50,000	50,00	0,01	1150,0	1150	12	1150,0	1150	12
230,00	230,0	1,1	20,000	20,00	0,12	0,9000	0,900	0,006	49,000	49,00	0,03	4140,0	4140	21	4600,0	4600	26

230,00	230,0	0,7	20,000	20,00	0,07	1,0000	1,000	0,003	49,000	49,00	0,02	4600,0	4600	18	4600,0	4600	19
230,00	230,0	1,4	1,000	1,00	0,06	0,8000	0,800	0,003	51,000	51,00	0,03	184,0	184	26	230,0	230	26
230,00	230,0	0,7	5,000	5,00	0,06	0,8000	0,800	0,003	51,000	51,00	0,01	920,0	920	15	1150,0	1150	11
230,00	230,0	1,4	20,000	20,00	0,11	0,8000	0,800	0,004	50,000	50,00	0,02	3680,0	3680	26	4600,0	4600	17
230,00	230,0	1,0	20,000	20,00	0,06	0,9000	0,900	0,003	50,000	50,00	0,03	4140,0	4140	23	4600,0	4600	14
230,00	230,0	1,3	1,000	1,00	0,11	0,9000	0,900	0,003	51,000	51,00	0,02	207,0	207	19	230,0	230	14
230,00	230,0	1,7	5,000	5,00	0,11	0,9000	0,900	0,004	51,000	51,00	0,03	1035,0	1035	17	1150,0	1150	30
230,00	230,0	0,6	20,000	20,00	0,07	1,0000	1,000	0,003	50,000	50,00	0,03	4600,0	4600	16	4600,0	4600	13
230,00	230,0	1,2	20,000	20,00	0,07	0,8000	0,800	0,003	51,000	51,00	0,03	3680,0	3680	23	4600,0	4600	29
230,00	230,0	1,4	1,000	1,00	0,10	1,0000	1,000	0,003	51,000	51,00	0,03	230,0	230	11	230,0	230	15
230,00	230,0	1,4	5,000	5,00	0,08	1,0000	1,000	0,005	51,000	51,00	0,02	1150,0	1150	28	1150,0	1150	28
230,00	230,0	1,4	20,000	20,00	0,12	0,9000	0,900	0,003	51,000	51,00	0,02	4140,0	4140	12	4600,0	4600	14
230,00	230,0	1,3	20,000	20,00	0,13	1,0000	1,000	0,003	51,000	51,00	0,03	4600,0	4600	20	4600,0	4600	11

Tra i risultati interessanti di questo lavoro, possiamo mettere quindi in evidenza ad esempio l'incertezza sulla misura di potenza attiva, così come illustrato nelle seguenti figure:

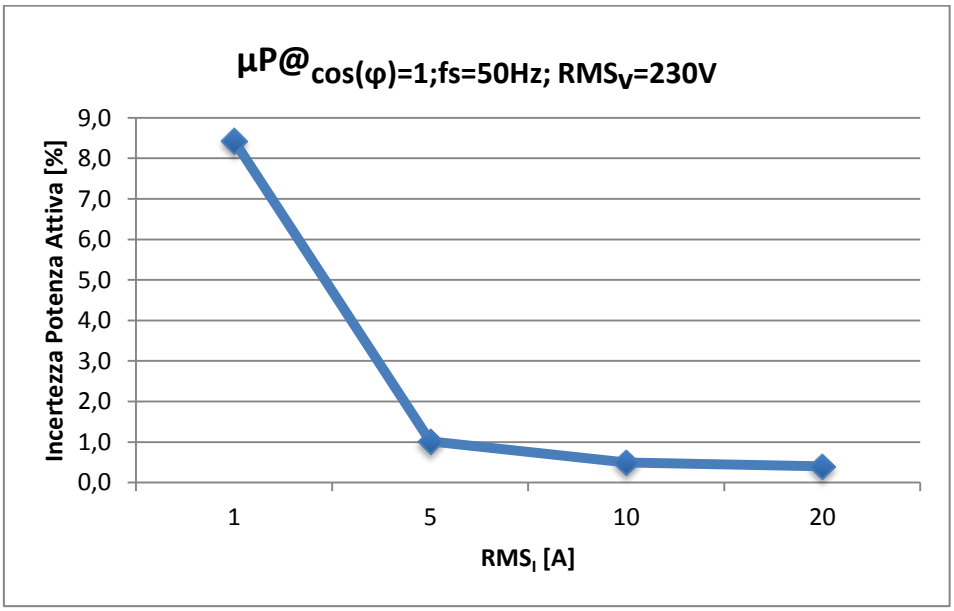


Figura 23: Incertezza nella misura di potenza rispetto alla corrente

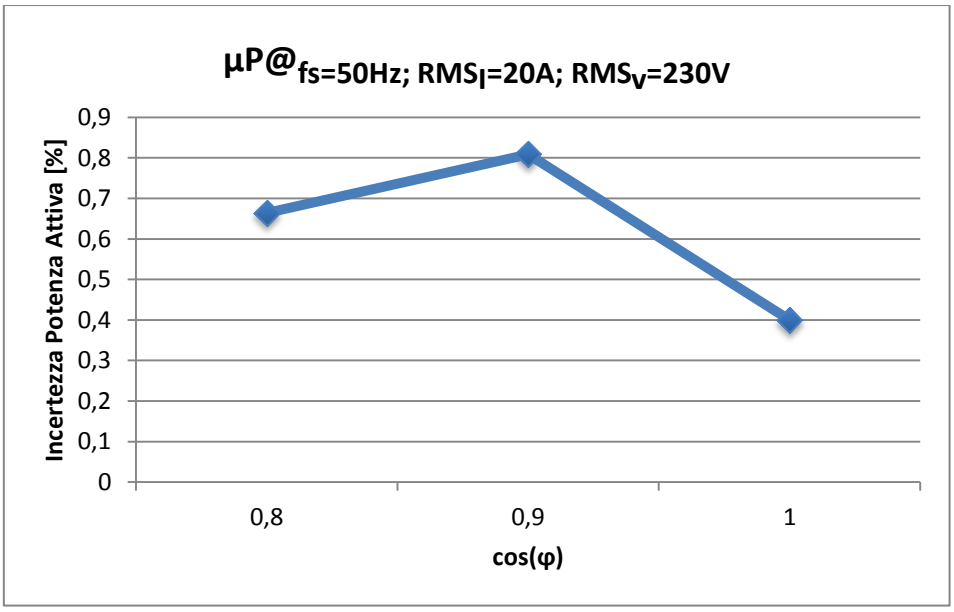


Figura 24: Incertezza nella misura di potenza rispetto al fattore di potenza

mentre invece la l'incertezza sulla misura del RMS_V risulta molto contenuta, così come illustrato nella figura:

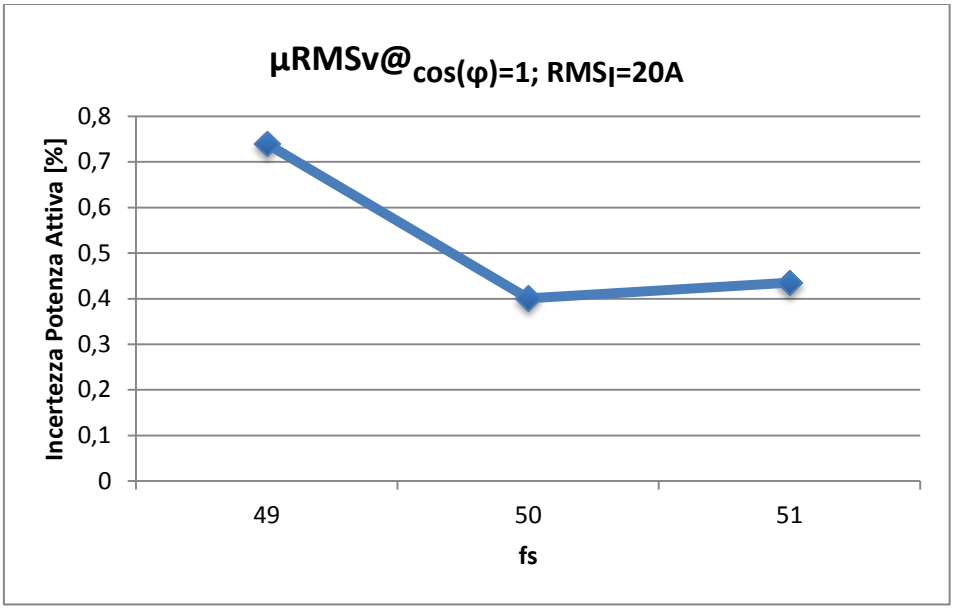


Figura 25: Incertezza nella misura di potenza rispetto alla frequenza

3 Confronto con un misuratore commerciale

La seguente analisi è stata condotta allo scopo di verificare le prestazioni degli SM implementati in reali condizioni di funzionamento. A tale scopo sono stati monitorati i consumi energetici di tipo elettrico che sono oggetto di tariffazione di un complesso commerciale ubicato nel comune di Aversa (NA). Più in precisamente è stato eseguito in maniera continuativa il monitoraggio dei consumi registrando ogni minuto gli assorbimenti medi in termini di potenza attiva e reattiva per una durata complessiva di circa otto giorni e, più precisamente, le registrazioni sono cominciate in data 17/07/2015 alle ore 16:30 circa e sono terminate il 25/07/2015 alle ore 18:00 circa. Durante questi giorni, tutte le apparecchiature servite sono state impiegate nelle modalità e nei tempi consueti tipici della normale attività lavorativa. Allo scopo del confronto le misurazioni sono state effettuate in parallelo sia da un prototipo dello SM implementato sia un misuratore di tipo commerciale meglio specificato nel seguito. Inoltre all'atto dell'avvio e dell'arresto della registrazione è stata presa nota della lettura dei contatori installati per la tariffazione dell'energia elettrica attiva e reattiva.

I sistema per la misura dell'energia attiva e dell'energia reattiva installati presso l'utenza in oggetto sono costituiti da due contatori elettromeccanici mostrati in appendice e i cui dati di targa sono riportati in tab.I.

Tab. I: Dati di targa dei gruppi di misura.

Energia Attiva	
Marca	Costruttori Associati Meridionali S.P.A.
Tipologia costruttiva	Contatore trifase a quattro fili
Modello	7CA 55 45
Portate	3x230 (400) V, 1.5-6 A, 50 Hz
Costante contatore	750 giri/kWh
Matricola	matr.08 003287

Energia Reattiva	
Marca	Costruttori Associati Meridionali S.P.A.
Tipologia costruttiva	Contatore trifase a quattro fili
Modello	7CB 55 45/90°
Portate	3x400 V, 1.5-6 A, 50 Hz
Costante contatore	750 giri/kWh
Matricola	matr.08 003292

I contatori sono alimentati da trasformatori di corrente installati in cabina.

Come strumentazione commerciale di confronto è stata impiegata un'apparecchiatura per l'analisi della qualità dell'energia elettrica, prodotta dalla Fluke, modello Memobox 808, dotata di un sistema di acquisizione analogico/digitale con risoluzione di 16 bit. Tale apparecchiatura esegue la misura dei principali indici che definiscono la qualità dell'alimentazione elettrica dei sistemi trifase secondo quanto prescritto dallo Standard 61000-4-30 una cui sintesi è riportata nella prima parte.

Di seguito si riporta la configurazione utilizzata del Memobox 808:

- PQ Log; Potenza
- Numero di serie:4/08
- Measurement code: Type A
- Periodo di Misura: 1 minuto
- Tensione Nominale: 400 V

In particolare per tale apparecchiatura è garantita un'incertezza intrinseca di 0,1% per la misura della tensione, minore del 2% per la misura della corrente con sonde flessibili, minore del 0,3 gradi per la misura della fase. L'accuratezza complessiva, pertanto, per le misure di potenza attiva e reattiva risulta senz'altro migliore del 3 %.

Con tale apparecchiatura, sono stati acquisiti i valori istantanei delle grandezze elencate d'interesse con una frequenza di campionamento di 10240 Hz (per un totale di 2048 punti a periodo della fondamentale a 50 Hz). Dai valori istantanei acquisiti sono stati calcolati le potenze elettriche registrando i valori minimi medi e massimi delle ogni minuto.

I collegamenti per la registrazione delle misure dei segnali di tensione e di corrente sono stati effettuati a valle dei contatori per la tariffazione ed a monte del quadro principale di alimentazione della sezione trifase dell'edificio con collegamento quadripolare a stella, come mostrato in figura. I trasduttori utilizzati per la corrente sono stati i 36-Inch Flexi Probe 4 Phase 3000/6000A Flex 4 e sono stati opportunamente tenuti in conto i fattori di scala introdotti.

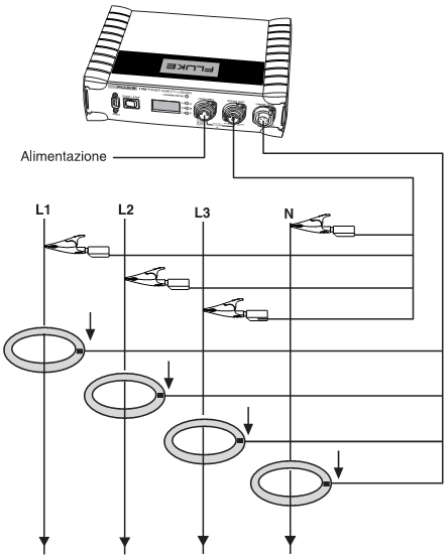


Figura 26: Inserzione del misuratore commerciale

3.1 Valutazione dei risultati di misura

Al termine della campagna di misurazione sono stati estratti i dati sia dallo MS cha dal memobox. I grafici dettagliati che riportano l'andamento delle grandezze oggetto delle analisi sono riportate in maniera esaustiva nel seguente paragrafo. In questo paragrafo verranno solamente riepilogati i risultati salienti del confronto per dimostrare le prestazioni in normali condizioni di funzionamento dello SM realizzato.

Sono di seguito riportate le figure che mostrano le statistiche sintetiche dei differenze, in termini percentuali, tra i risultati di misura ottenuti con l'apparecchiatura commerciale utilizzata e il prototipo presentato. I risultati sono accettabili ed in line con le attese.

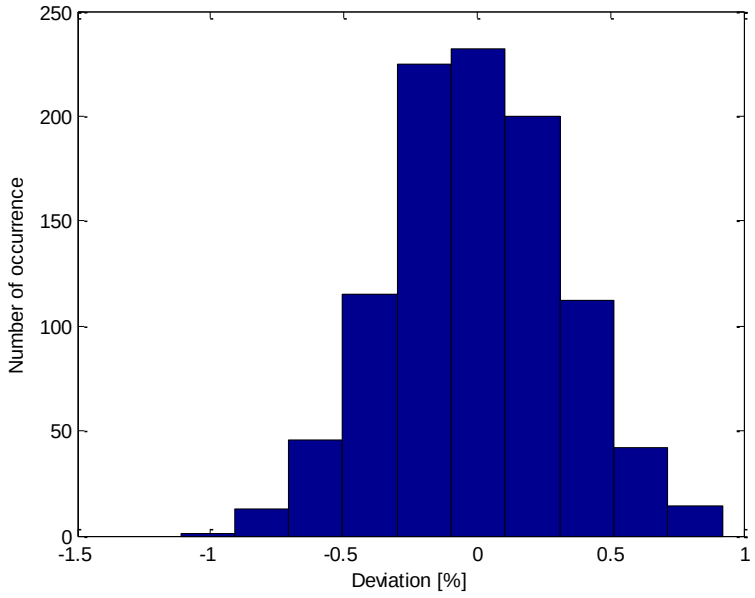


Figura 27: Scostamenti percentuali nelle misure di Potenza Attiva

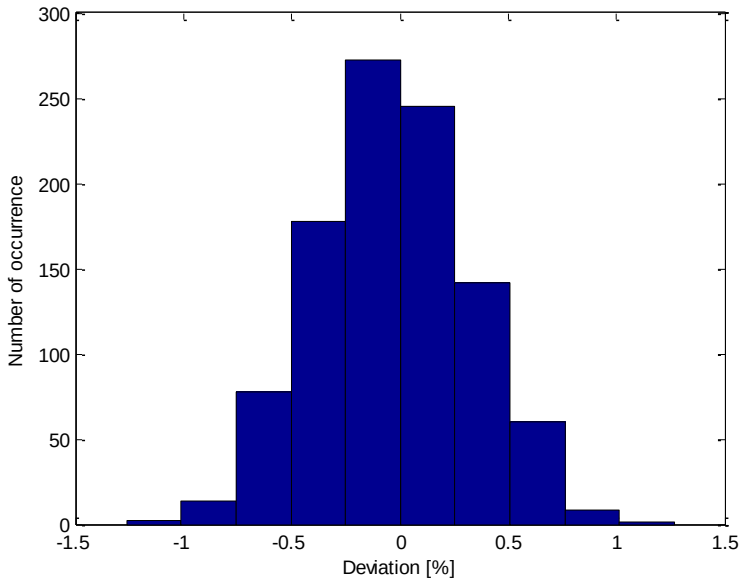


Figura 28: Scostamenti percentuali nelle misure di Potenza Reattiva

I valori delle letture riscontrate per i contatori di energia attiva e reattiva sono riportate in tabella 29 assieme al valore dell'incremento di lettura.

Tabella 29: Confronto misure d'energia

	Energia Attiva	Energia Reattiva
Energia misurata SM	696 kWh	576 kVARh
Energia misurata Memobox	689 kWh	567 kVARh
Scostamento	+7 kWh	+9 kVARh
Scostamento relativo	+1.0 %	+1.6 %

I valori misurati dal contatore di energia reattiva sono compatibili con quelli ottenuti dalla strumentazione utilizzata.

3.2 Grafici Dettagliati

Nel seguito sono riportati i grafici degli assorbimenti di energia attiva e reattiva che sono stati registrati durante le rilevazioni.

3.2.1 Grafici degli assorbimenti di potenza attiva

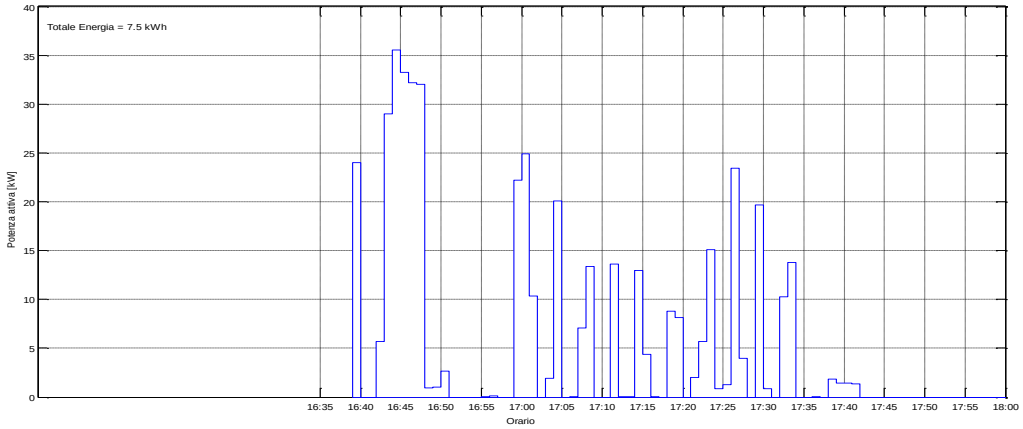


Figura 30: Potenza Attiva misurata il 17/07/2015 dalle ore 16:35

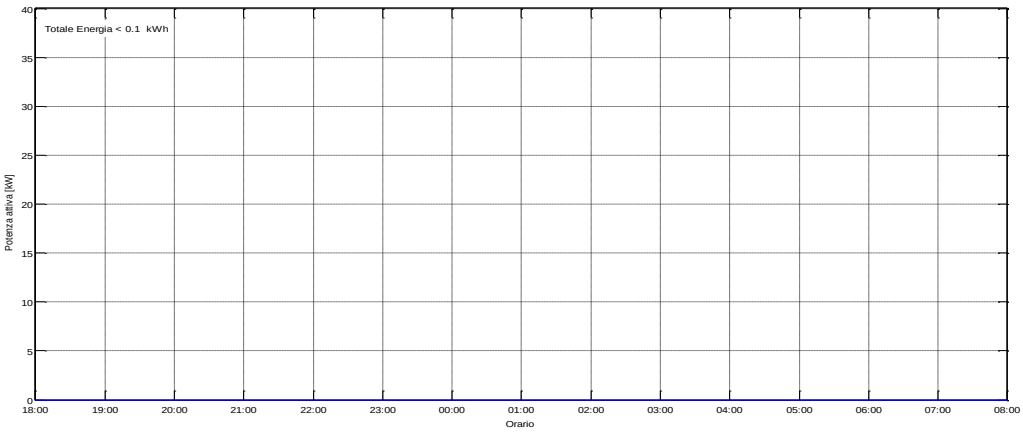


Figura 31: Potenza Attiva misurata il 17/07/2015 dalle ore 18:00

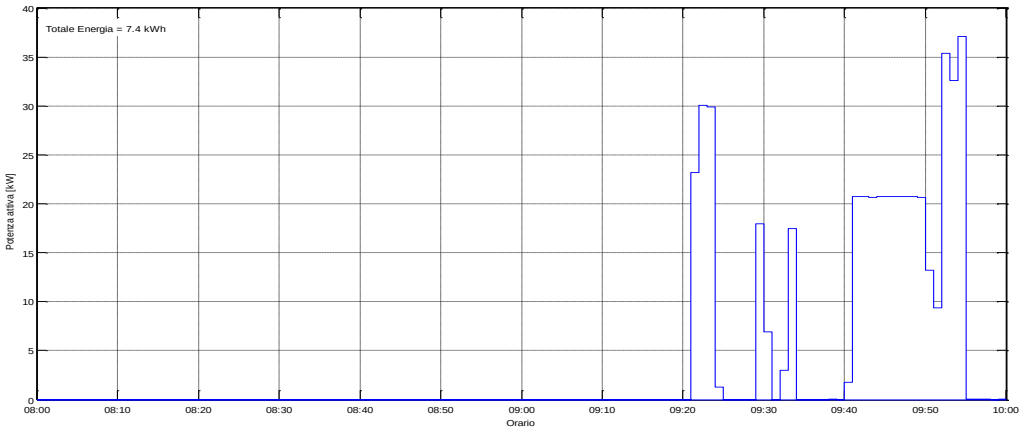


Figura 32: Potenza Attiva misurata il 18/07/2015 dalle ore 08:00

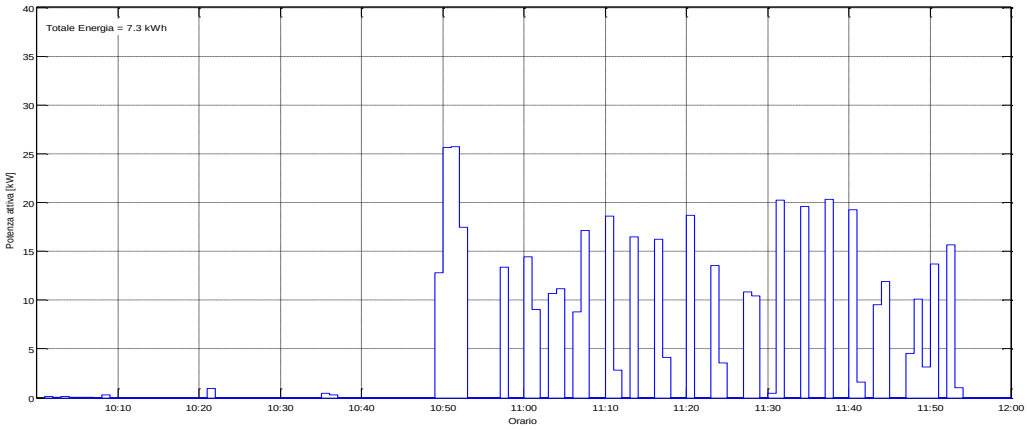


Figura 33: Potenza Attiva misurata il 18/07/2015 dalle ore 10:00

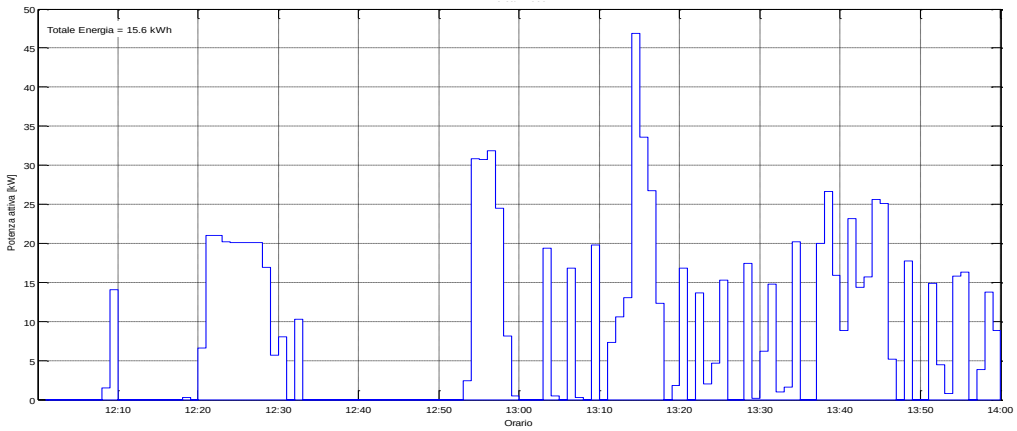


Figura 34: Potenza Attiva misurata il 18/07/2015 dalle ore 12:00

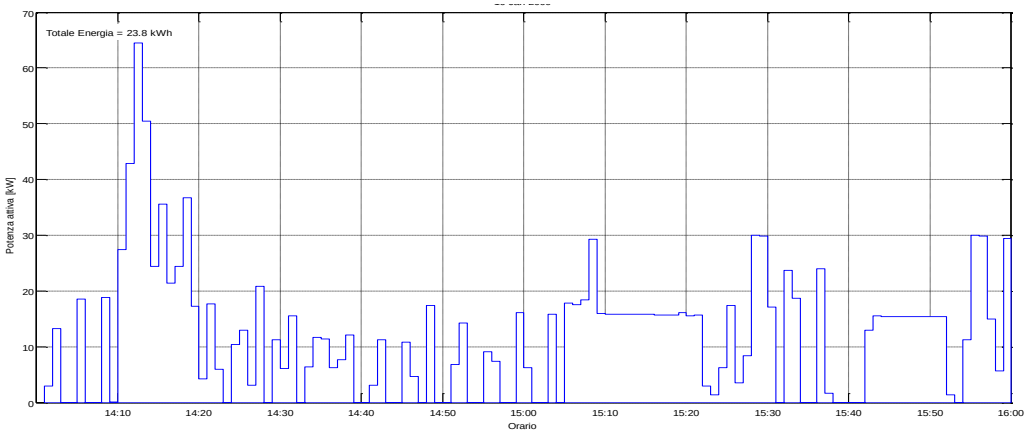


Figura 35: Potenza Attiva misurata il 18/07/2015 dalle ore 14:00

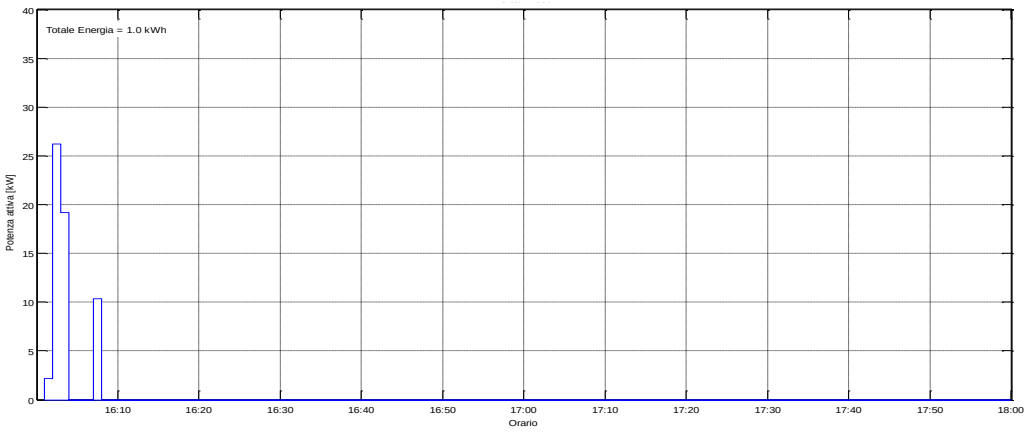


Figura 36: Potenza Attiva misurata il 18/07/2015 dalle ore 16:00

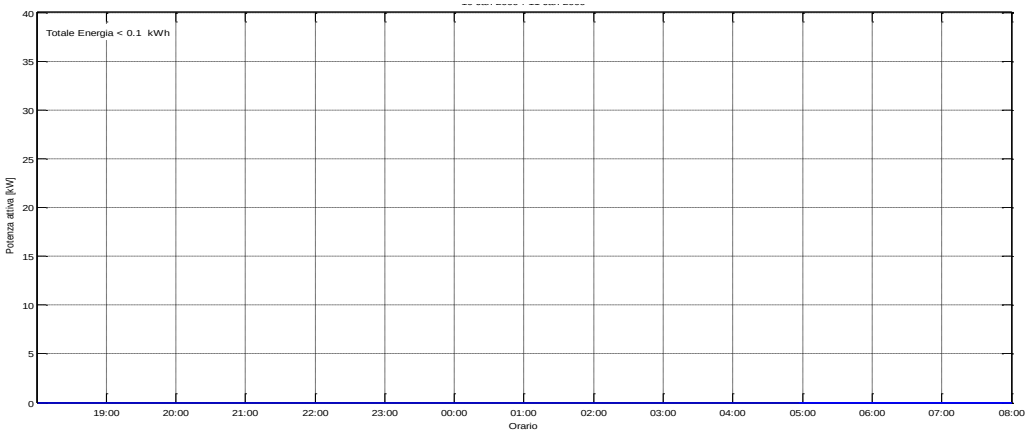


Figura 37: Potenza Attiva misurata il 18/07/2015 dalle ore 19:00

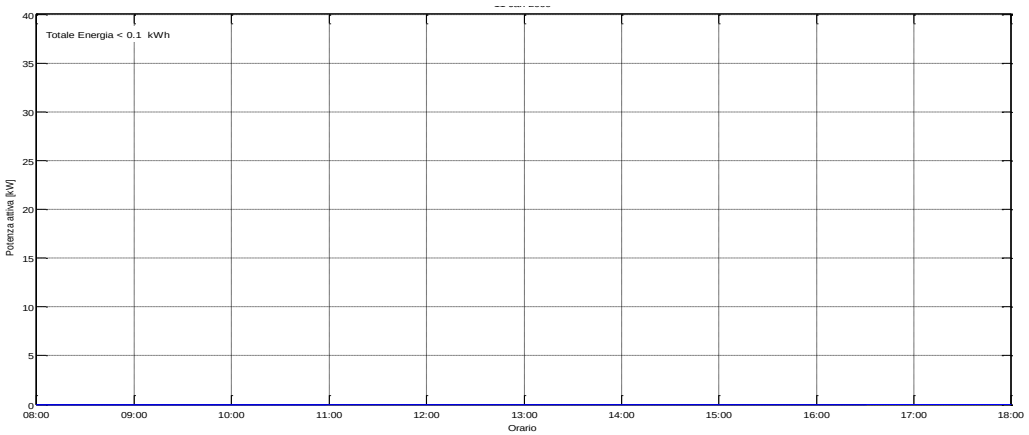


Figura 38: Potenza Attiva misurata il 19/07/2015 dalle ore 08:00

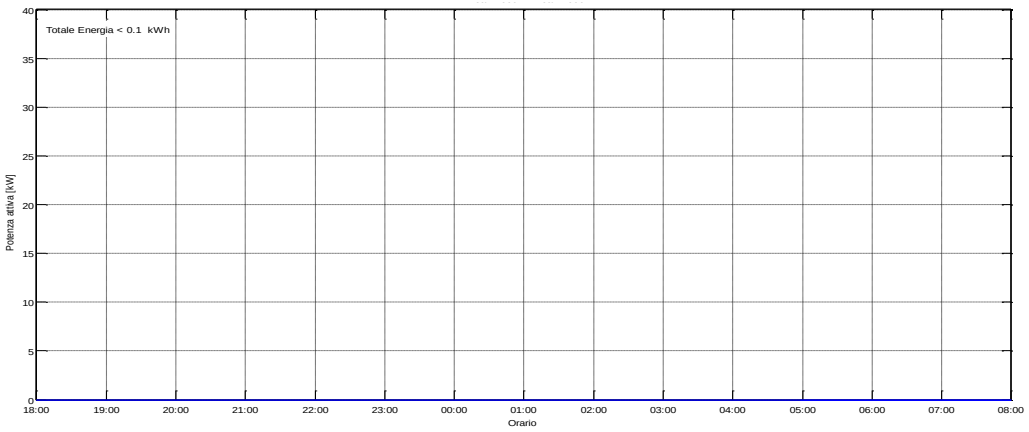


Figura 39: Potenza Attiva misurata il 19/07/2015 dalle ore 18:00

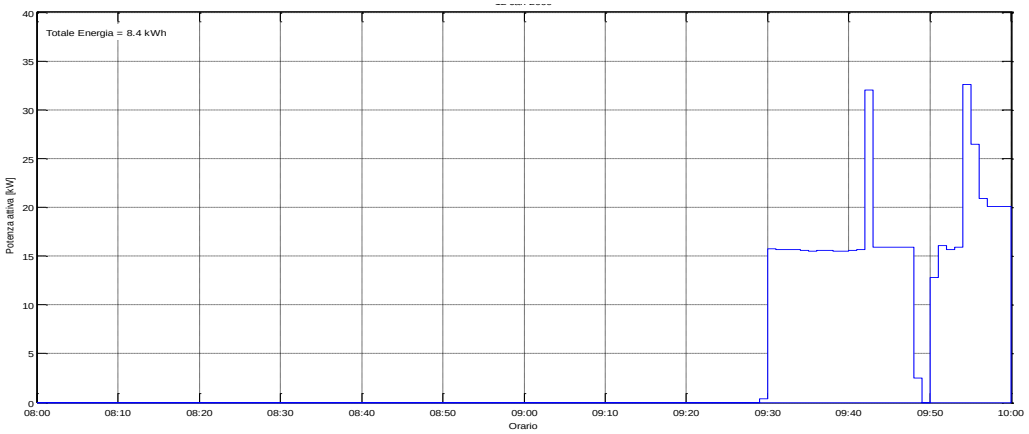


Figura 40: Potenza Attiva misurata il 20/07/2015 dalle ore 08:00

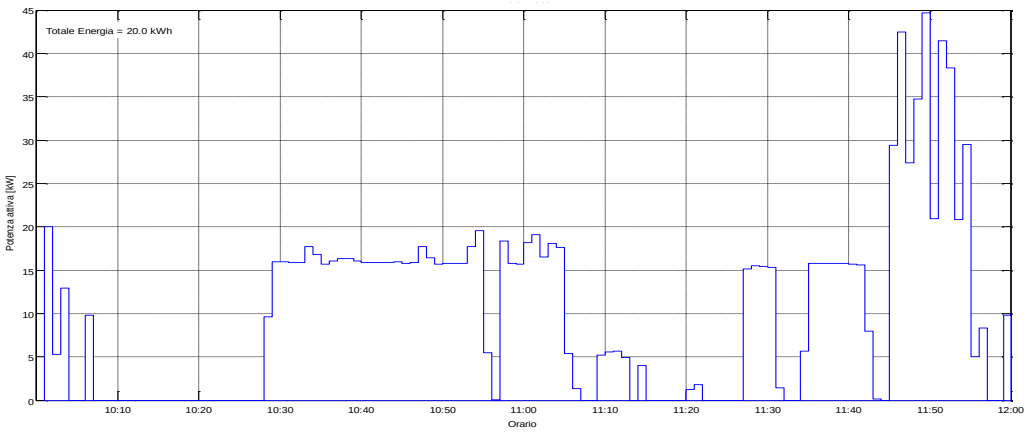


Figura 41: Potenza Attiva misurata il 20/07/2015 dalle ore 10:00

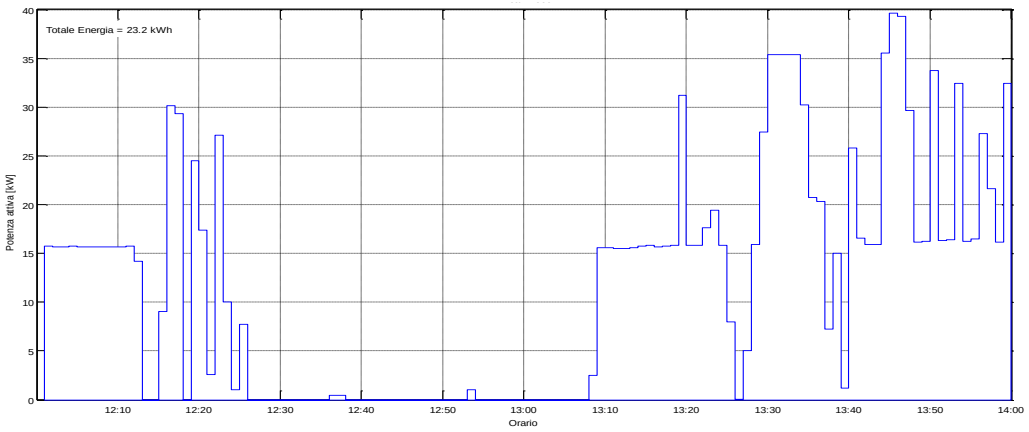


Figura 42: Potenza Attiva misurata il 20/07/2015 dalle ore 12:00

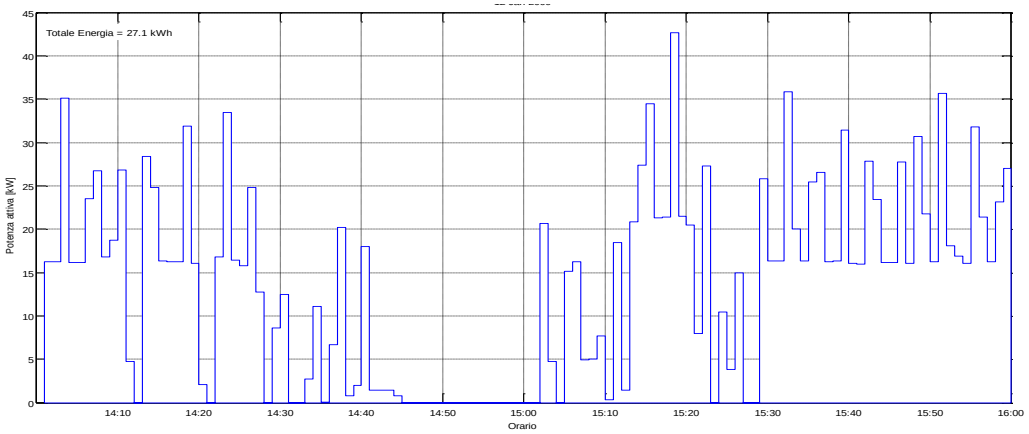


Figura 43: Potenza Attiva misurata il 20/07/2015 dalle ore 14:00

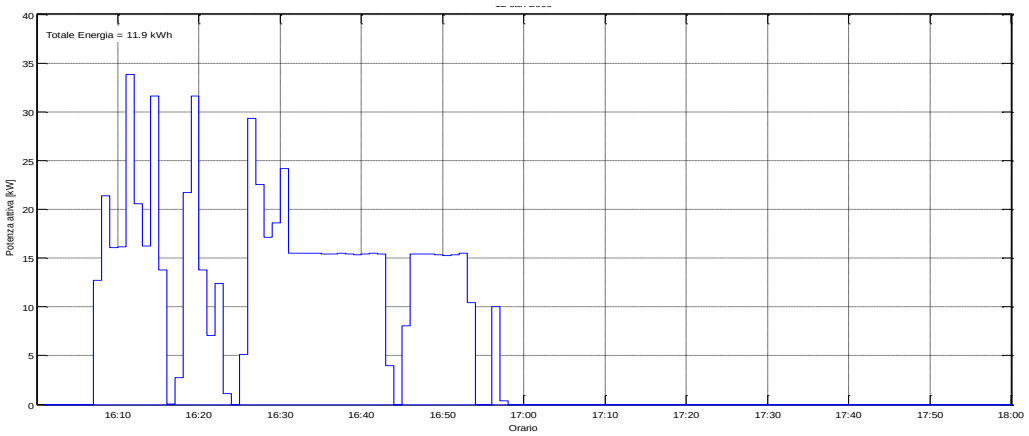


Figura 44: Potenza Attiva misurata il 20/07/2015 dalle ore 16:00

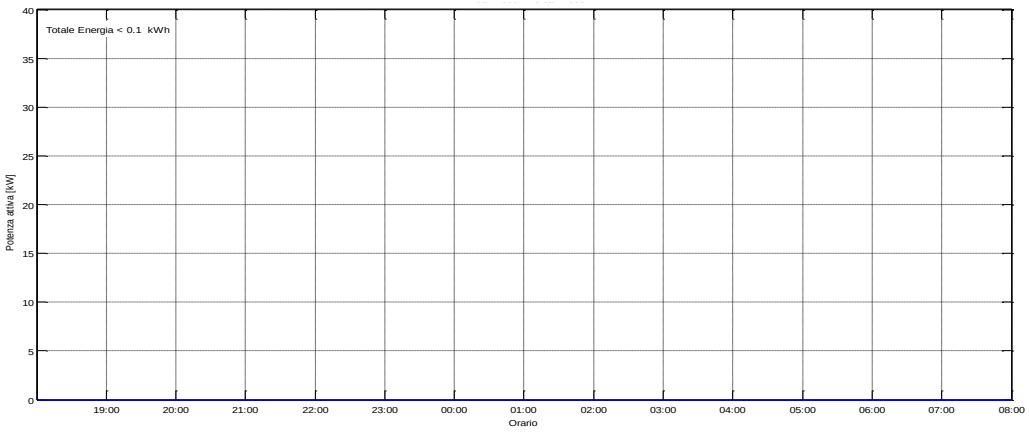


Figura 45: Potenza Attiva misurata il 20/07/2015 dalle ore 19:00

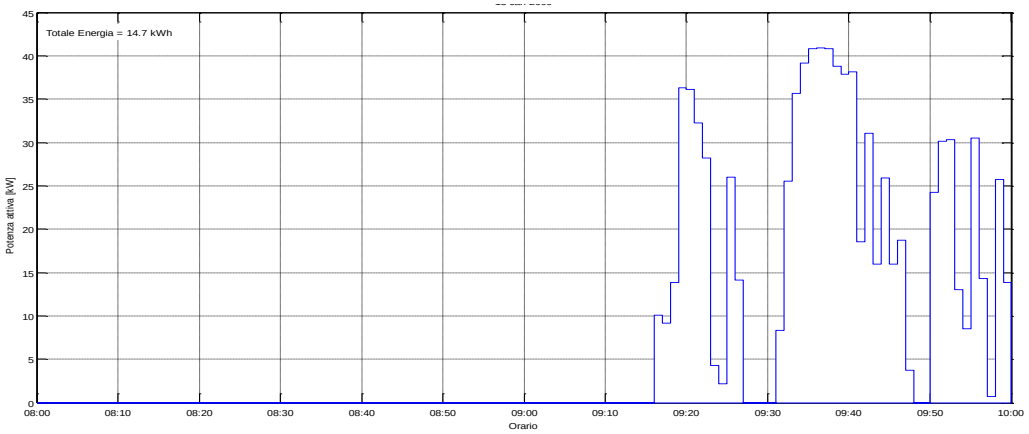
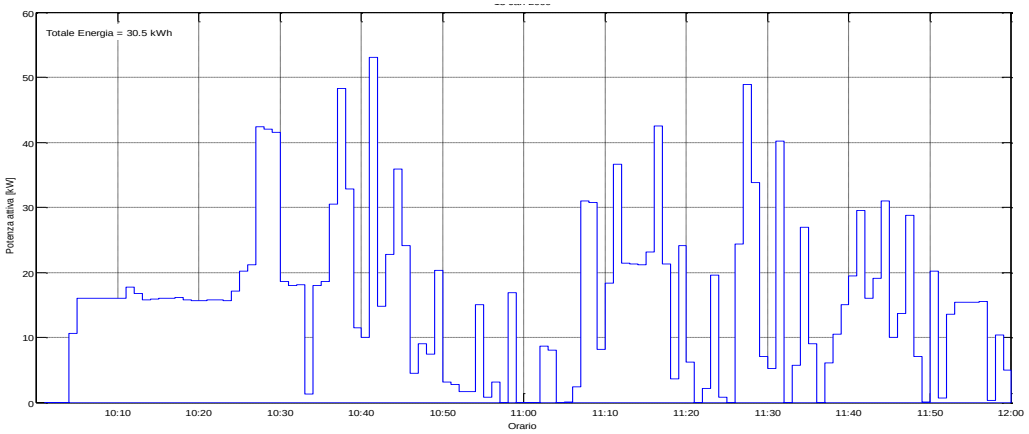


Figura 46: Potenza Attiva misurata il 21/07/2015 dalle ore 08:00



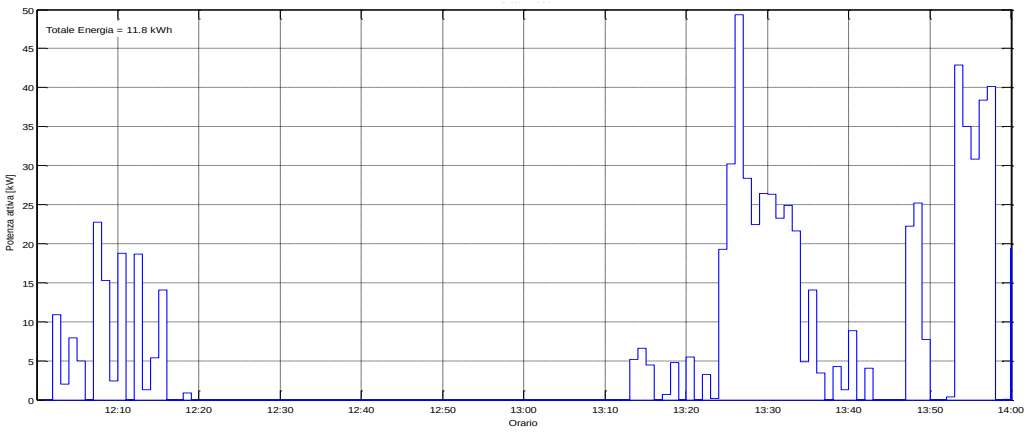


Figura 47: Potenza Attiva misurata il 21/07/2015 dalle ore 12:00

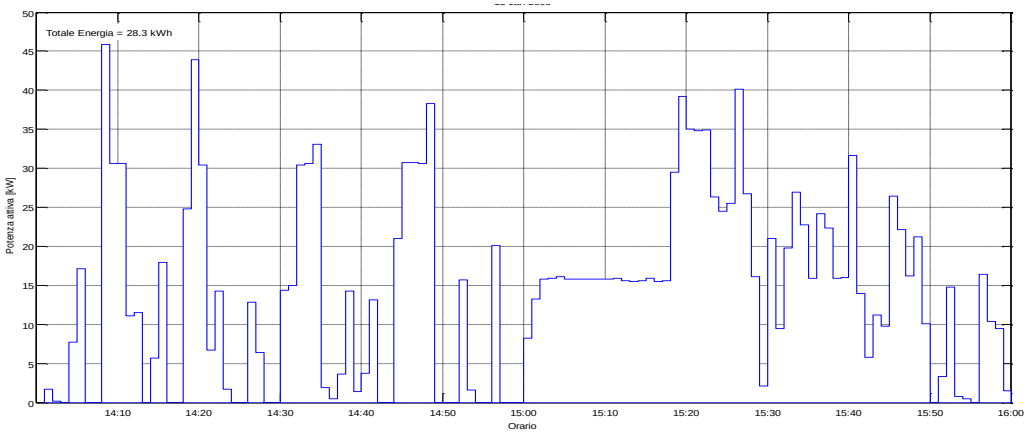


Figura 48: Potenza Attiva misurata il 21/07/2015 dalle ore 14:00

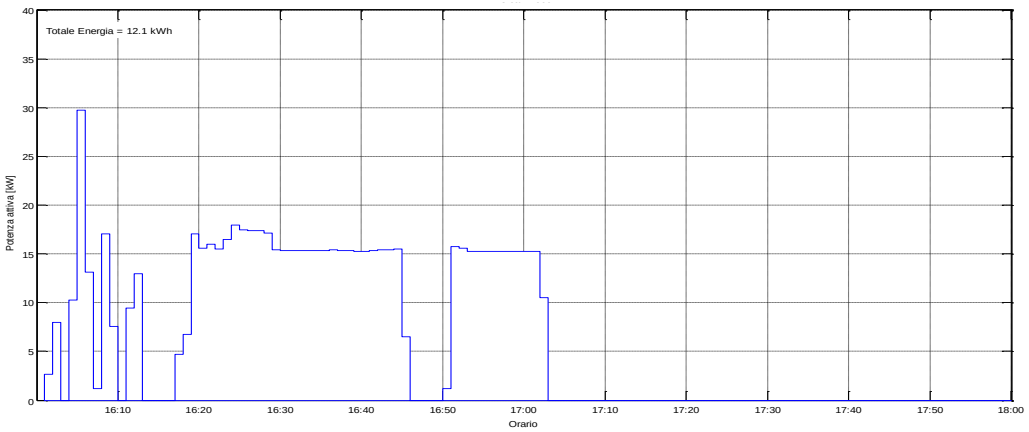


Figura 49: Potenza Attiva misurata il 21/07/2015 dalle ore 16:00

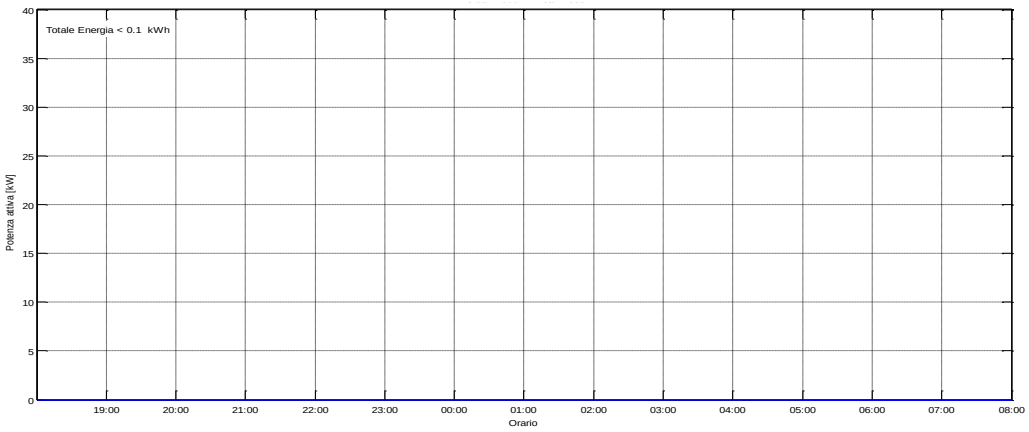


Figura 50: Potenza Attiva misurata il 21/07/2015 dalle ore 19:00

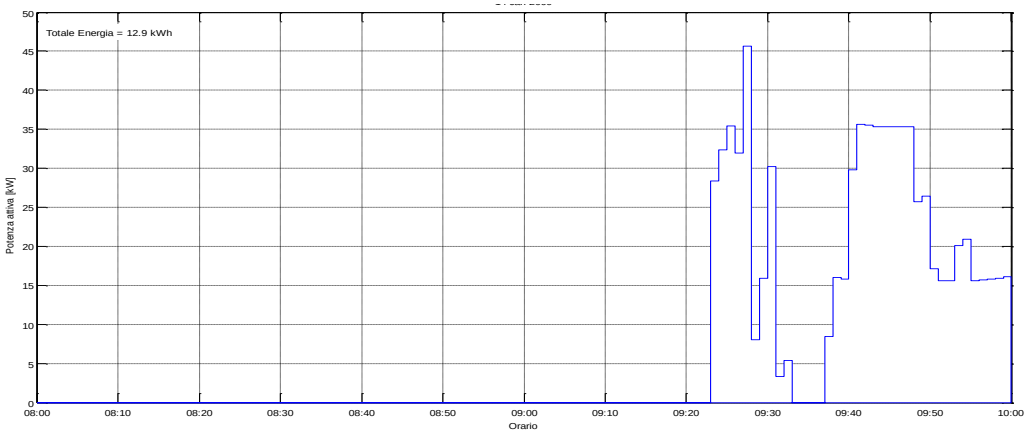


Figura 51: Potenza Attiva misurata il 22/07/2015 dalle ore 08:00

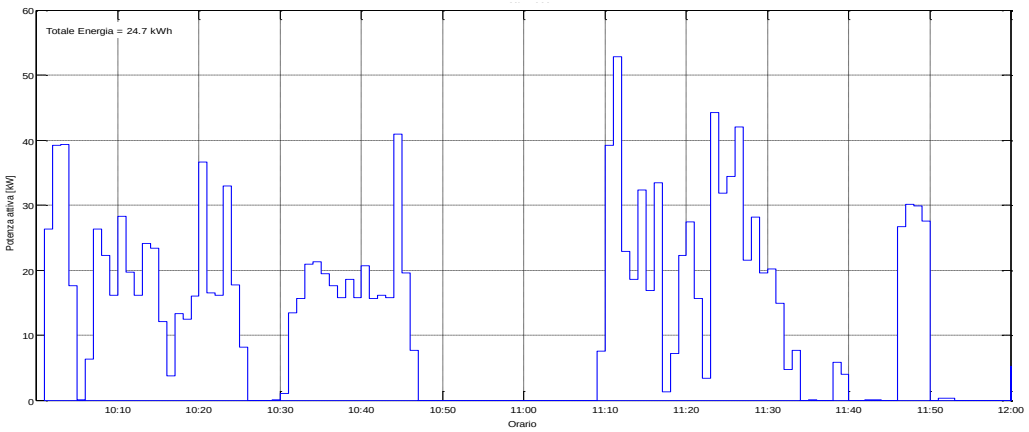


Figura 52: Potenza Attiva misurata il 22/07/2015 dalle ore 10:00

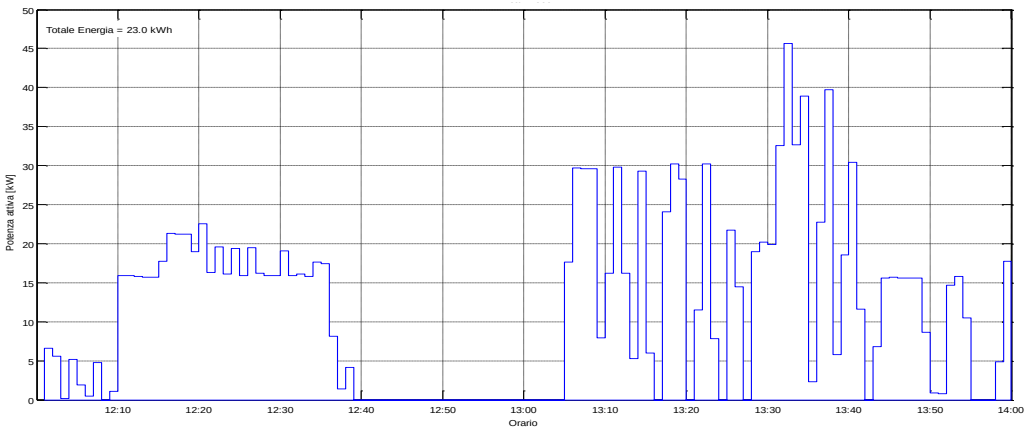


Figura 53: Potenza Attiva misurata il 22/07/2015 dalle ore 12:00

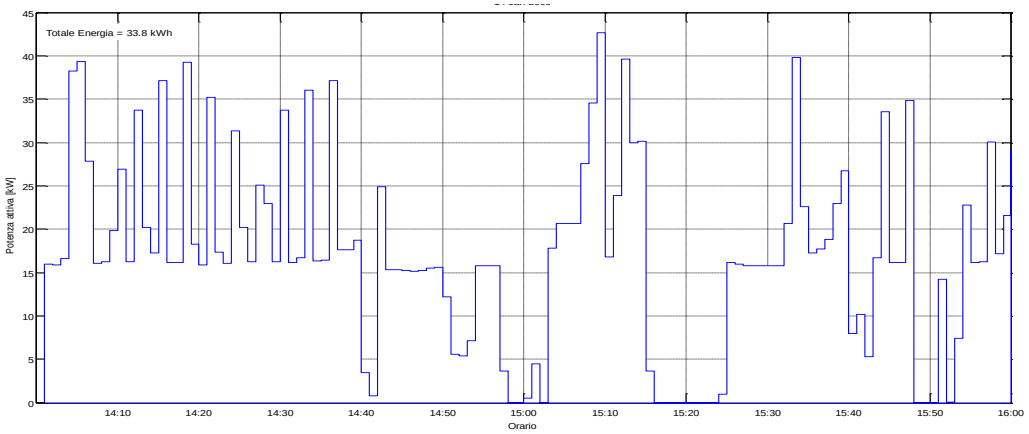


Figura 54: Potenza Attiva misurata il 22/07/2015 dalle ore 14:00

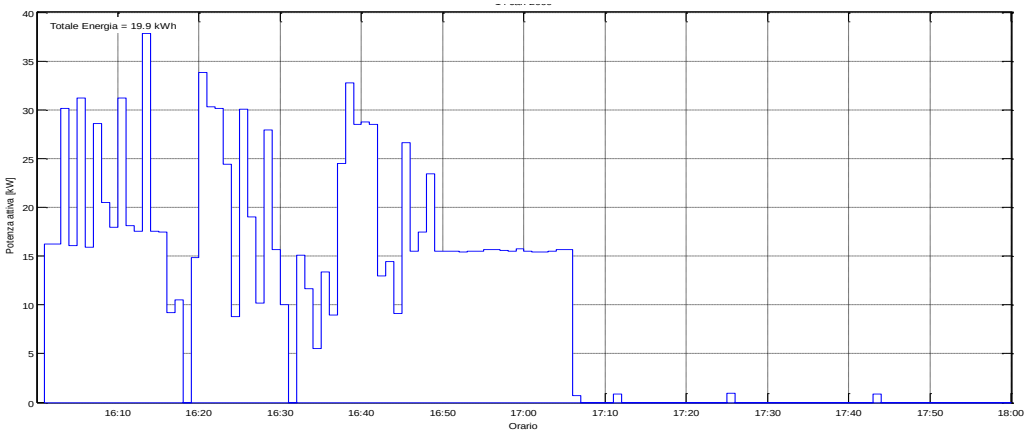


Figura 55: Potenza Attiva misurata il 22/07/2015 dalle ore 16:00

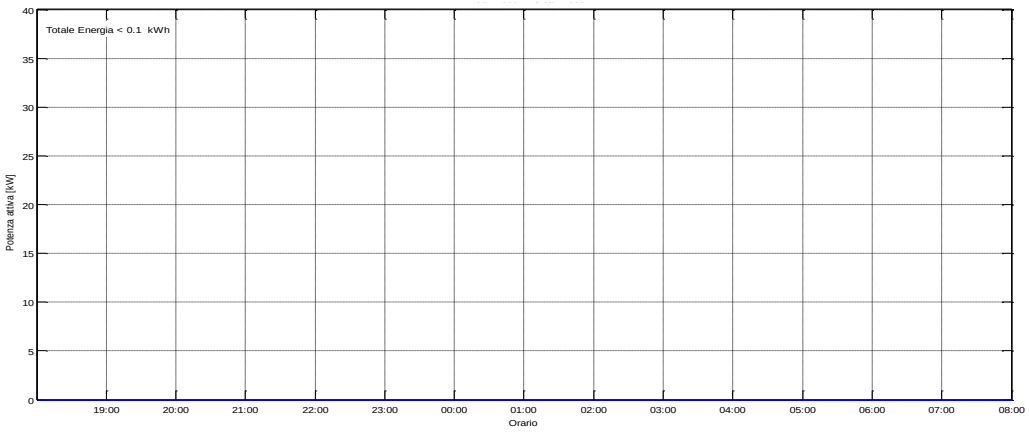


Figura 56: Potenza Attiva misurata il 22/07/2015 dalle ore 19:00

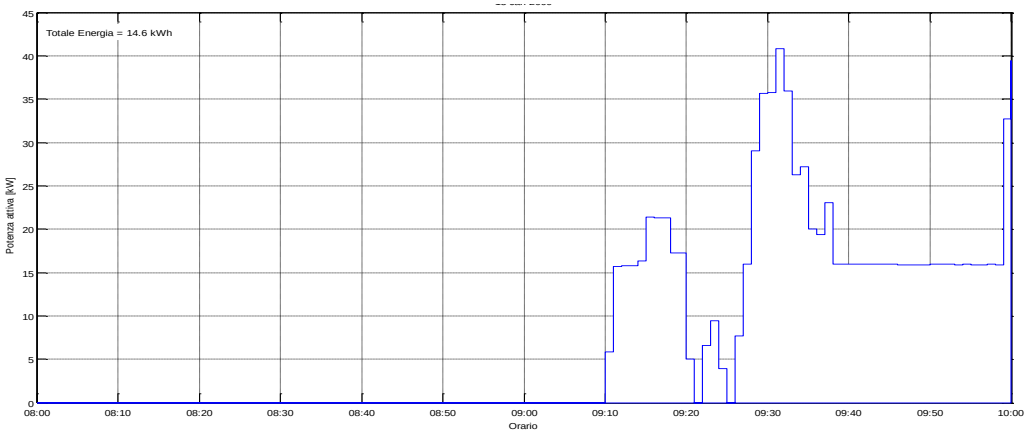


Figura 57: Potenza Attiva misurata il 23/07/2015 dalle ore 08:00

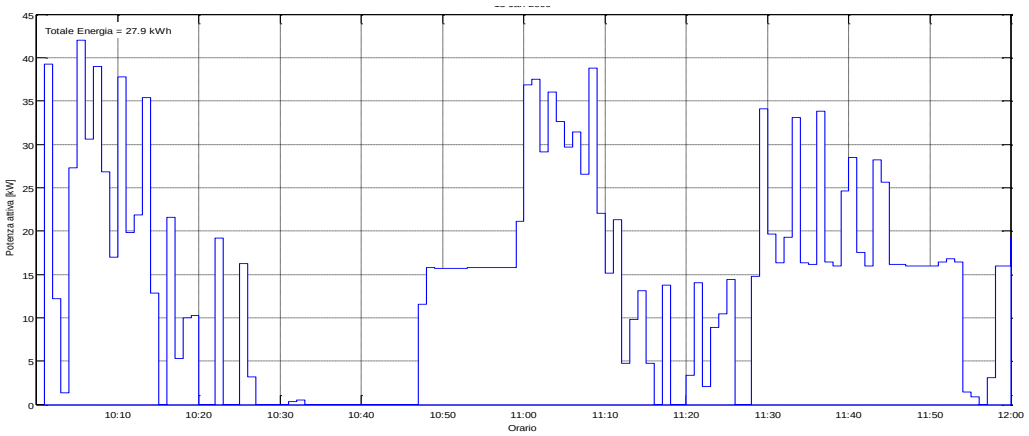


Figura 58: Potenza Attiva misurata il 23/07/2015 dalle ore 10:00

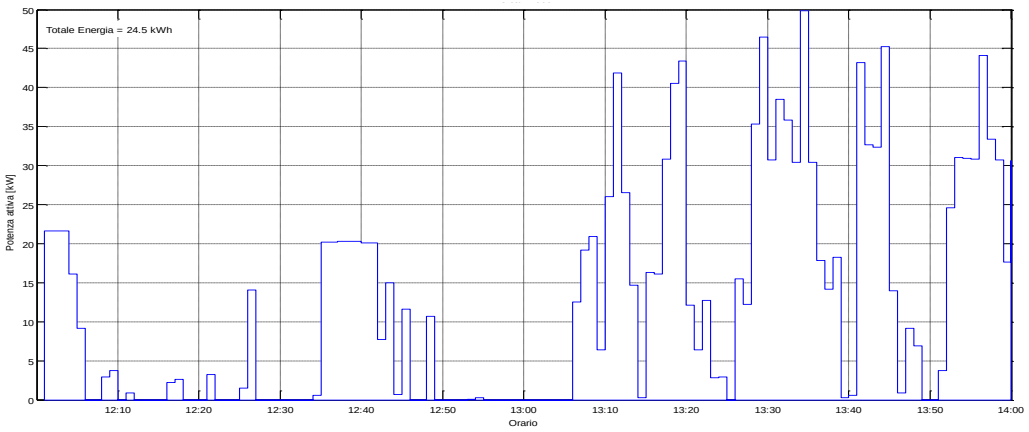


Figura 59: Potenza Attiva misurata il 23/07/2015 dalle ore 12:00

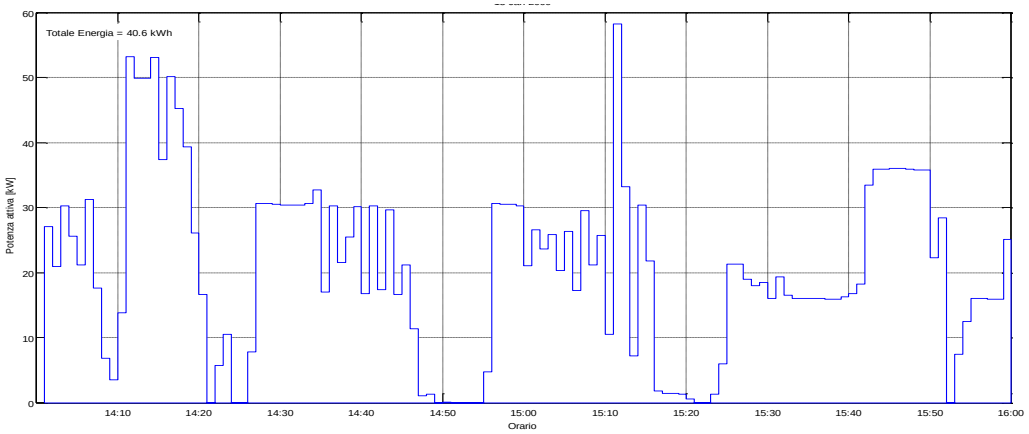


Figura 60: Potenza Attiva misurata il 23/07/2015 dalle ore 14:00

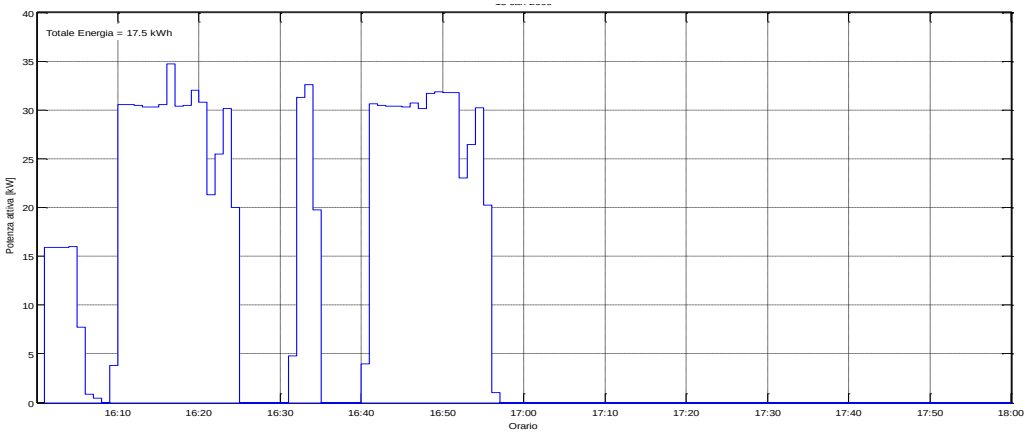


Figura 61: Potenza Attiva misurata il 23/07/2015 dalle ore 16:00

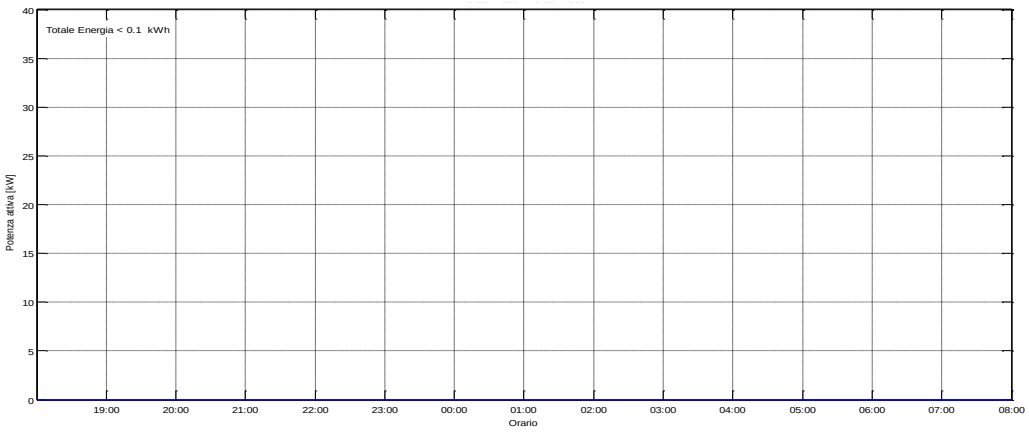


Figura 62: Potenza Attiva misurata il 23/07/2015 dalle ore 19:00

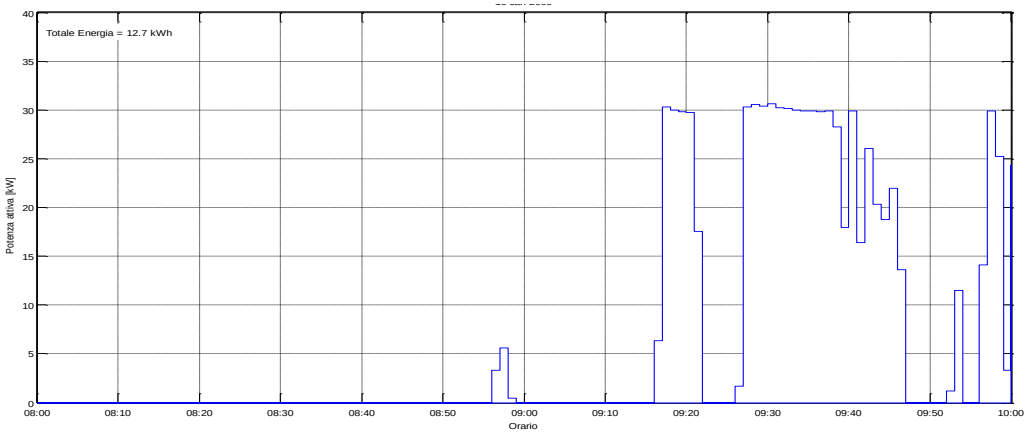


Figura 63: Potenza Attiva misurata il 24/07/2015 dalle ore 08:00

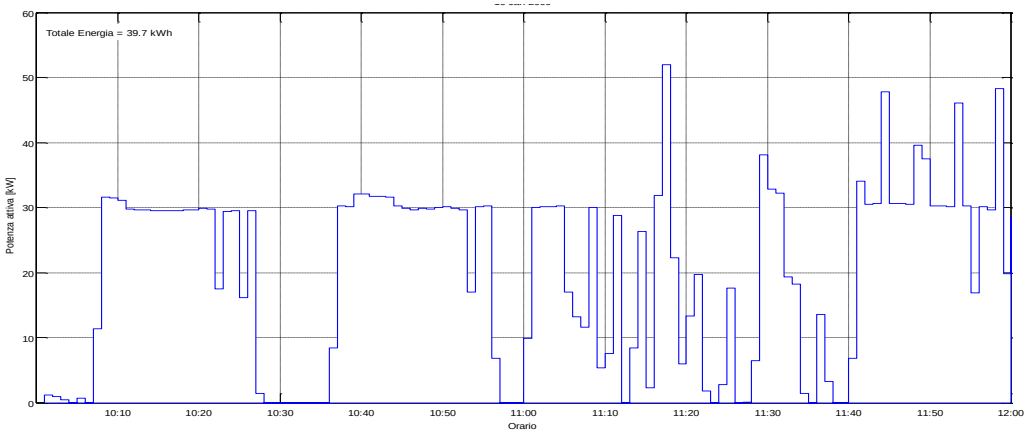


Figura 64: Potenza Attiva misurata il 24/07/2015 dalle ore 10:00

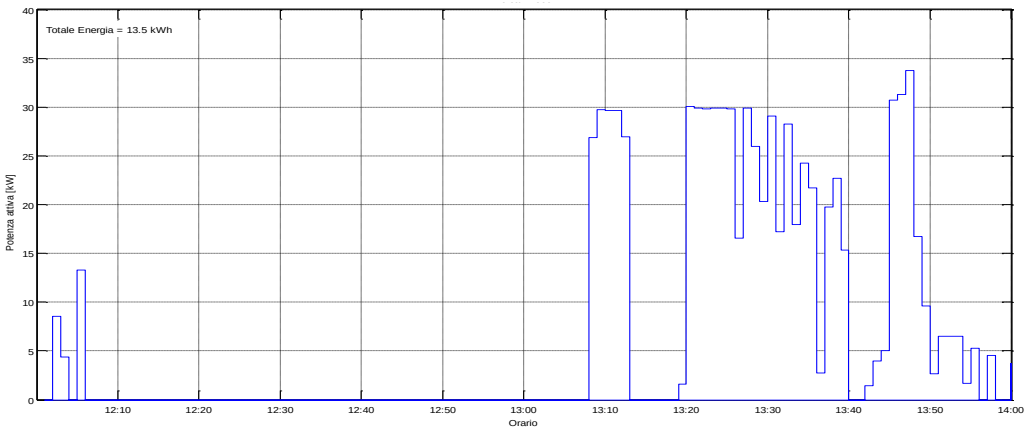


Figura 65: Potenza Attiva misurata il 24/07/2015 dalle ore 12:00

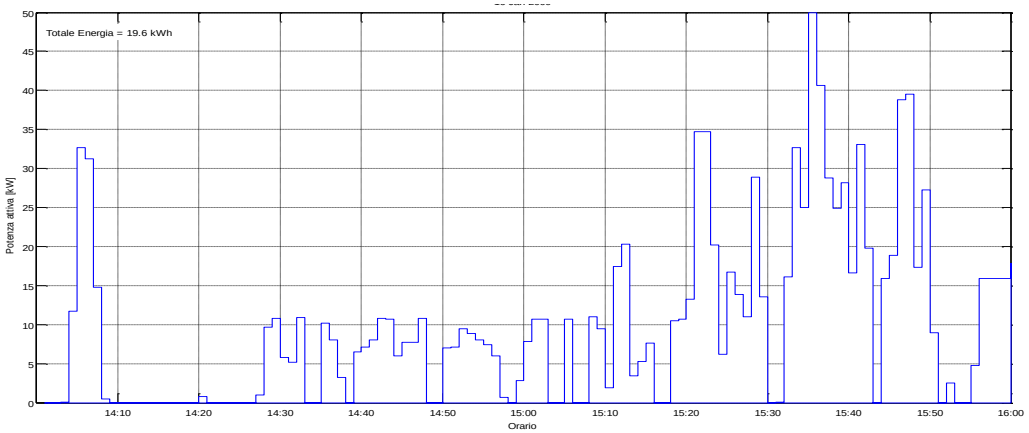


Figura 66: Potenza Attiva misurata il 24/07/2015 dalle ore 14:00

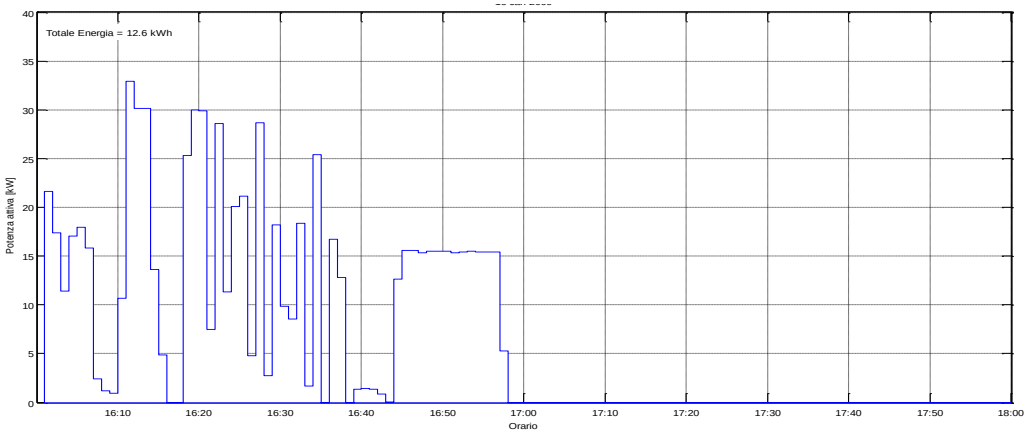


Figura 67: Potenza Attiva misurata il 24/07/2015 dalle ore 16:00

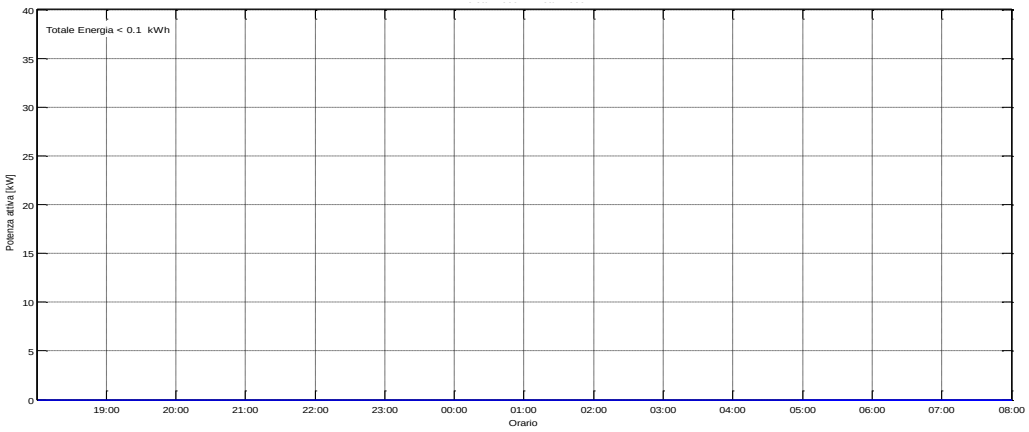


Figura 68: Potenza Attiva misurata il 24/07/2015 dalle ore 19:00

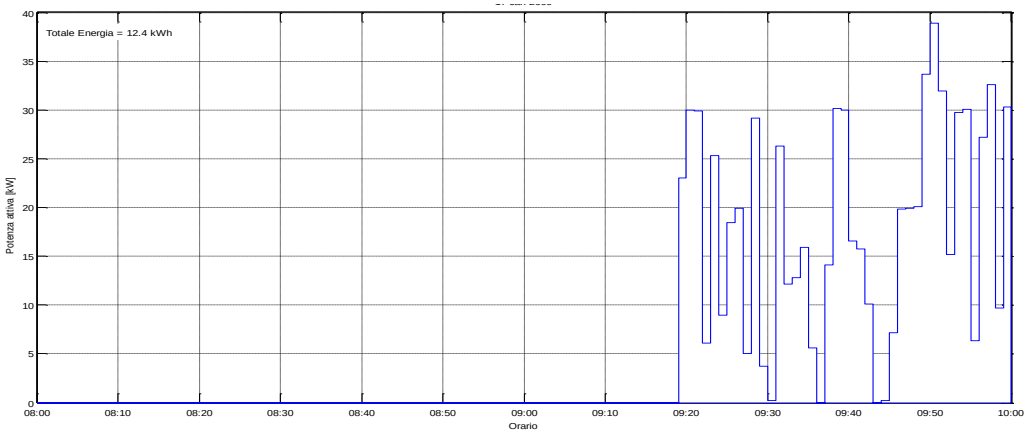


Figura 69: Potenza Attiva misurata il 25/07/2015 dalle ore 08:00

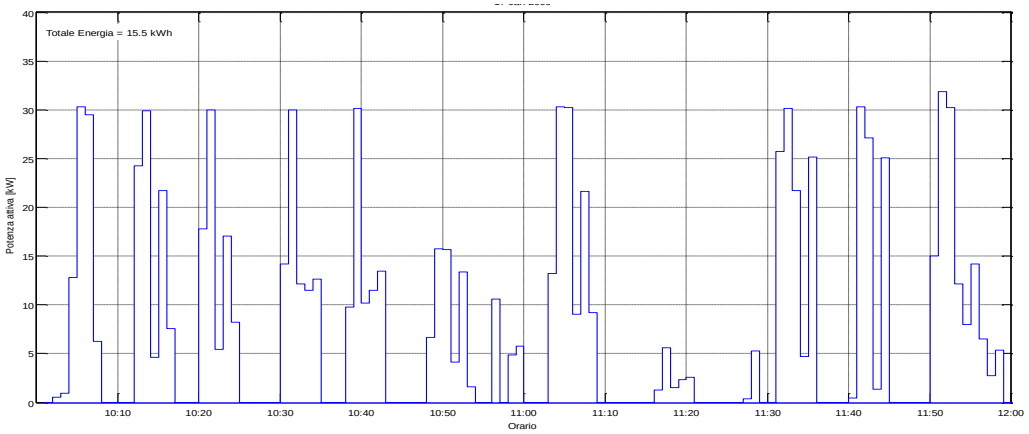


Figura 70: Potenza Attiva misurata il 25/07/2015 dalle ore 10:00

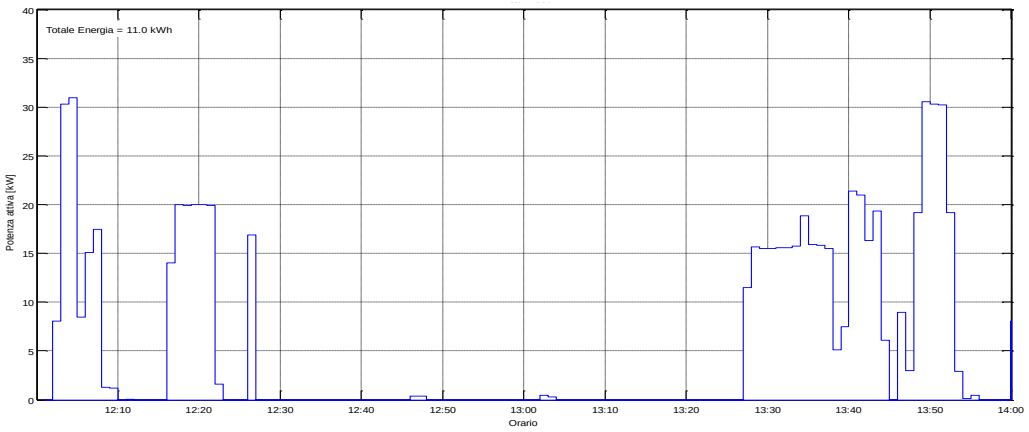


Figura 71: Potenza Attiva misurata il 25/07/2015 dalle ore 12:00

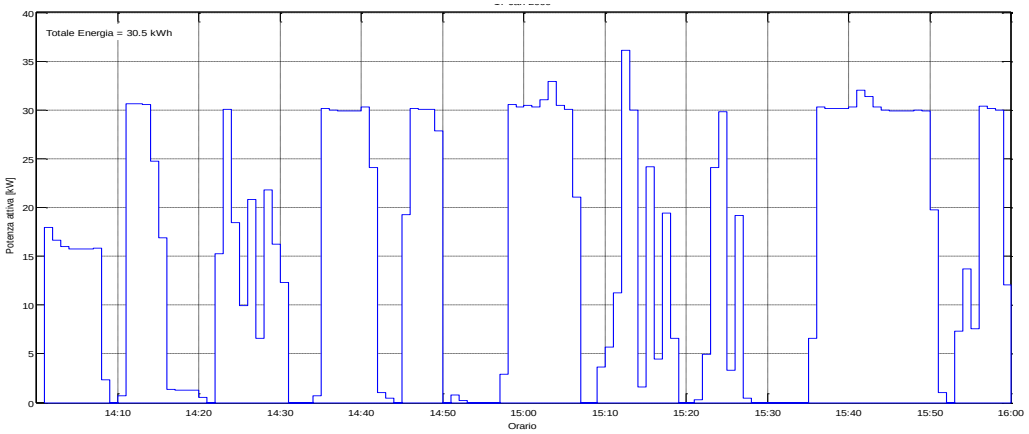


Figura 72: Potenza Attiva misurata il 25/07/2015 dalle ore 14:00

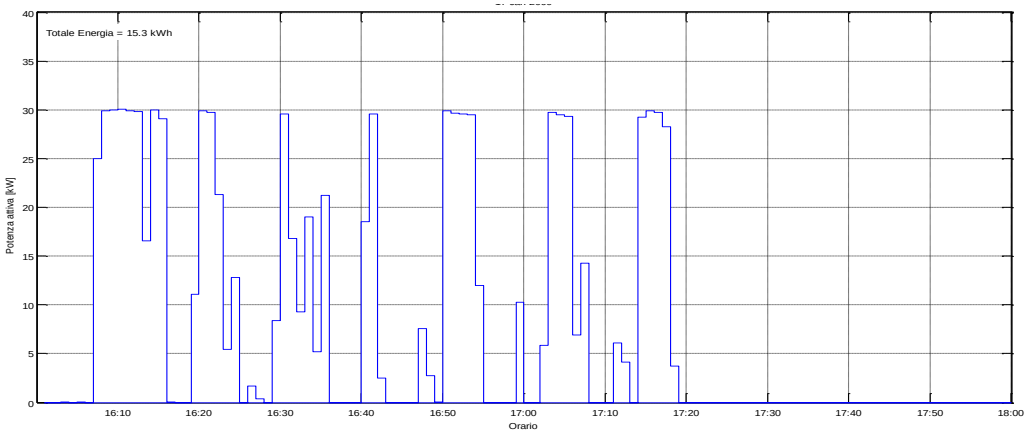


Figura 73: Potenza Attiva misurata il 25/07/2015 dalle ore 16:00

3.2.2 Grafici degli assorbimenti di potenza reattiva

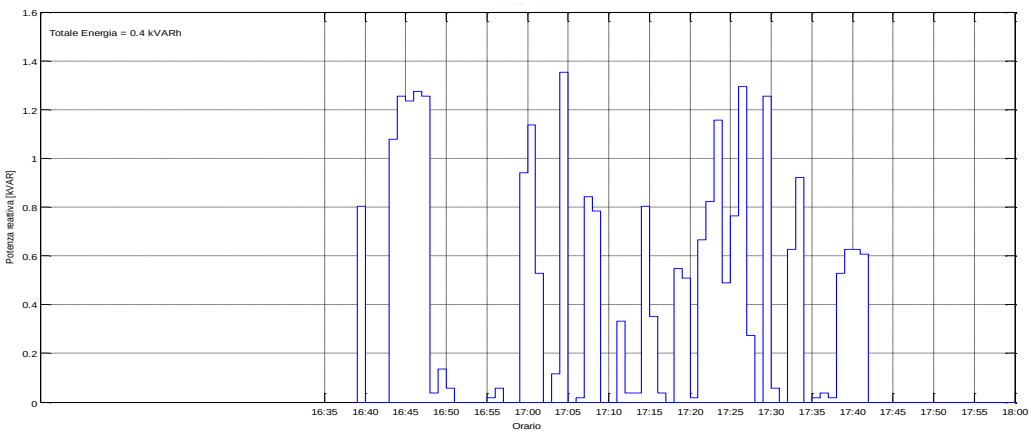


Figura 74: Potenza Reattiva misurata il 17/07/2015 dalle ore 16:35

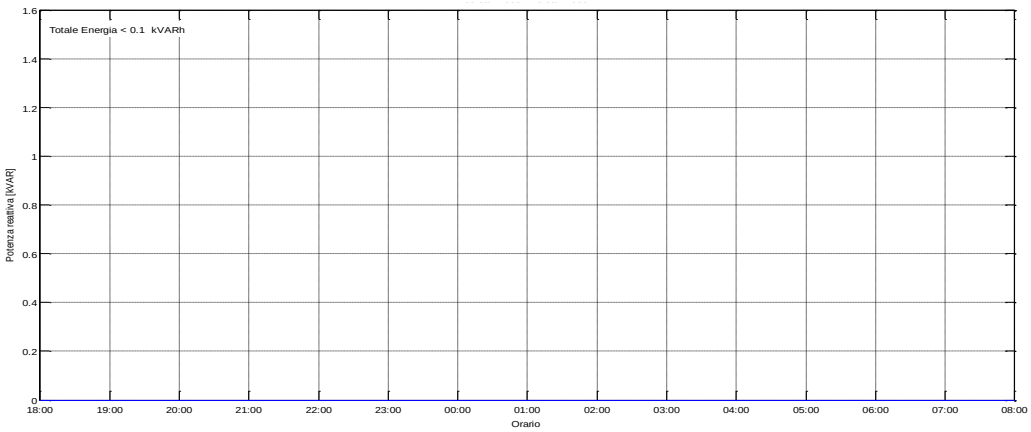


Figura 75: Potenza Reattiva misurata il 17/07/2015 dalle ore 18:00

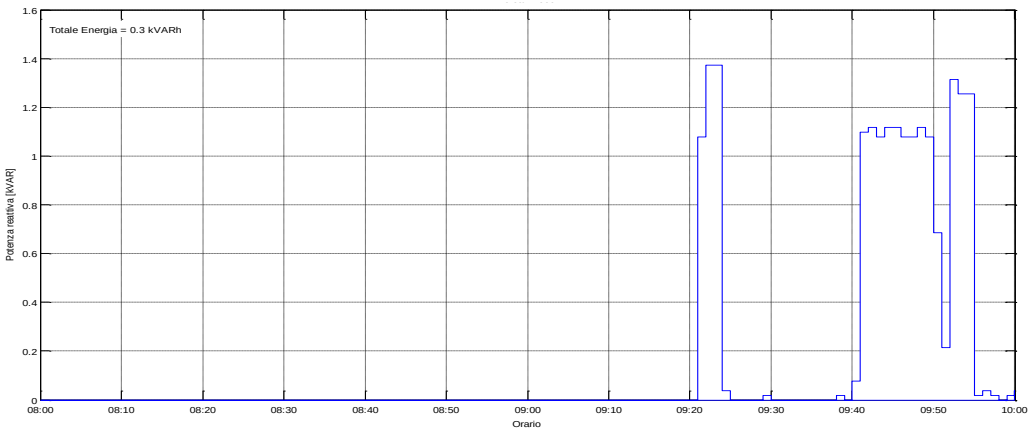


Figura 76: Potenza Reattiva misurata il 18/07/2015 dalle ore 08:00

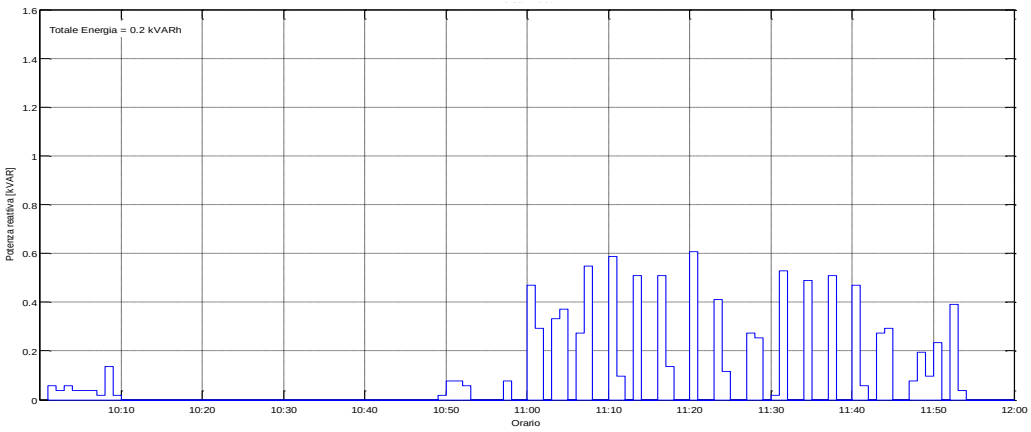


Figura 77: Potenza Reattiva misurata il 18/07/2015 dalle ore 10:00

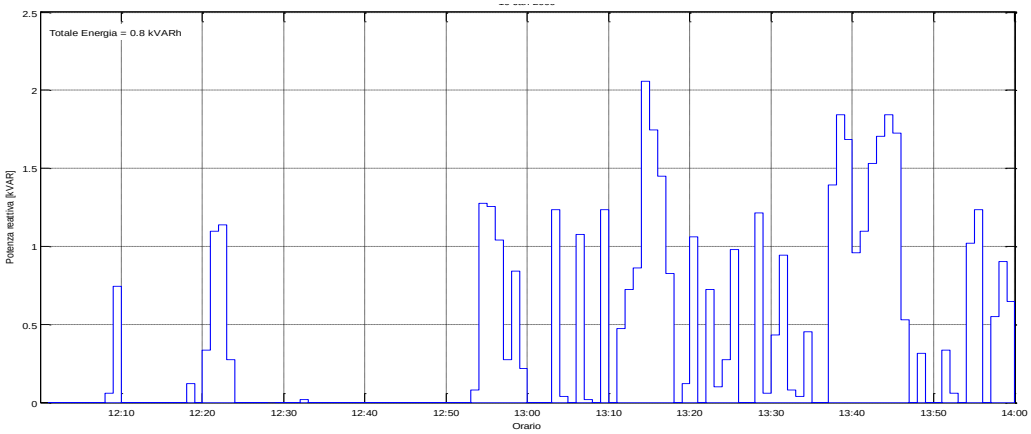


Figura 78: Potenza Reattiva misurata il 18/07/2015 dalle ore 12:00

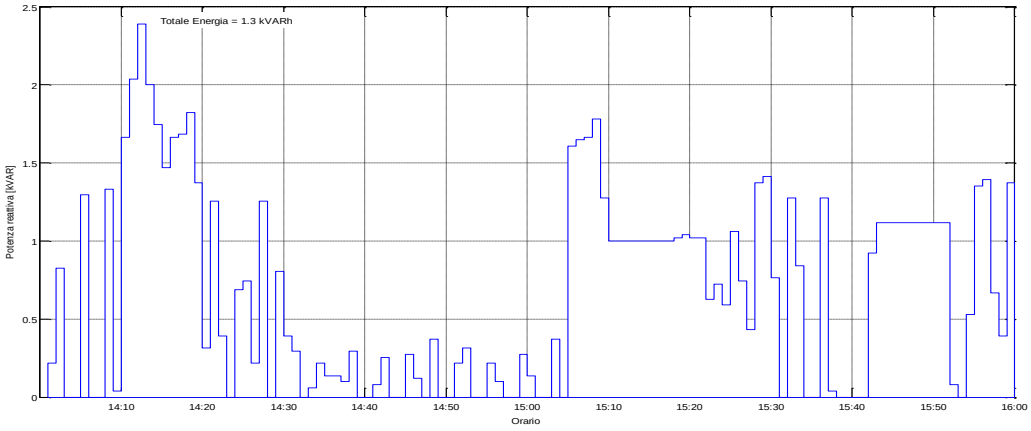


Figura 79: Potenza Reattiva misurata il 18/07/2015 dalle ore 14:00

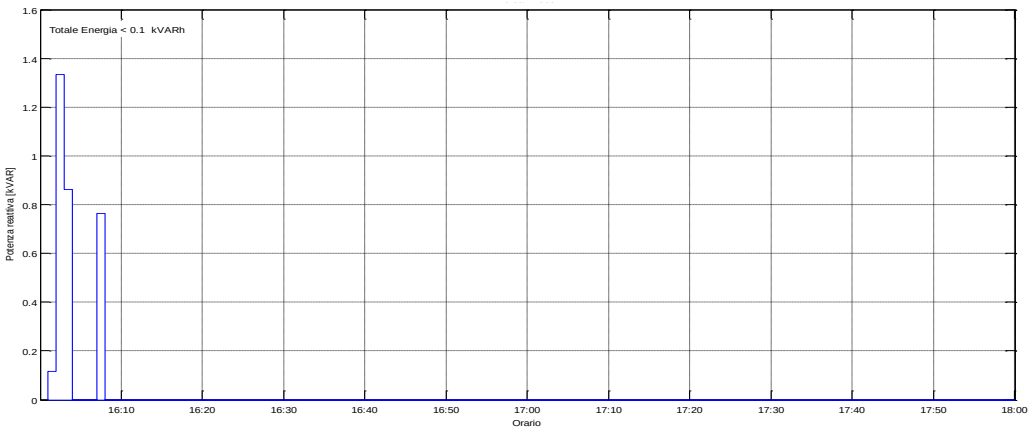


Figura 80: Potenza Reattiva misurata il 18/07/2015 dalle ore 16:00

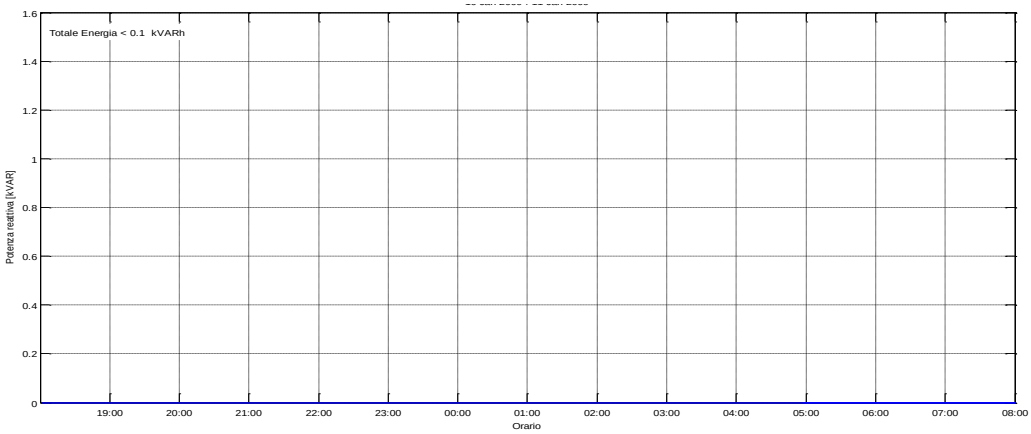


Figura 81: Potenza Reattiva misurata il 18/07/2015 dalle ore 19:00

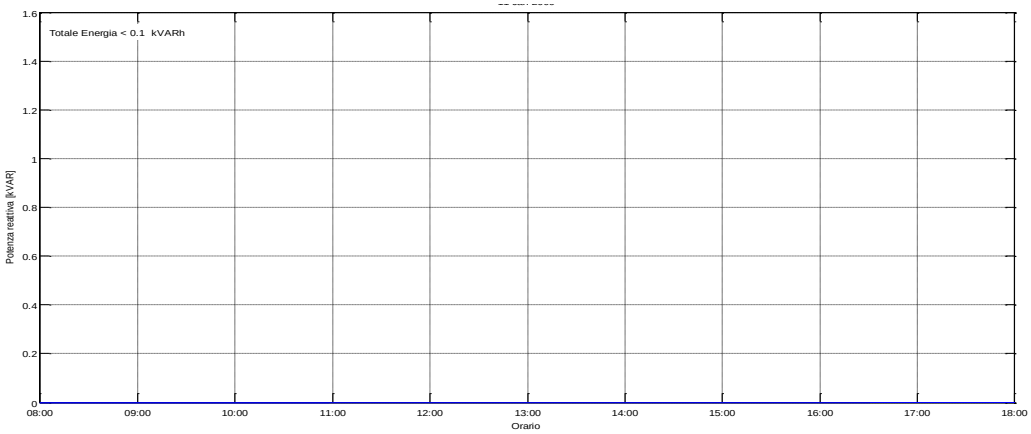


Figura 82: Potenza Reattiva misurata il 19/07/2015 dalle ore 08:00

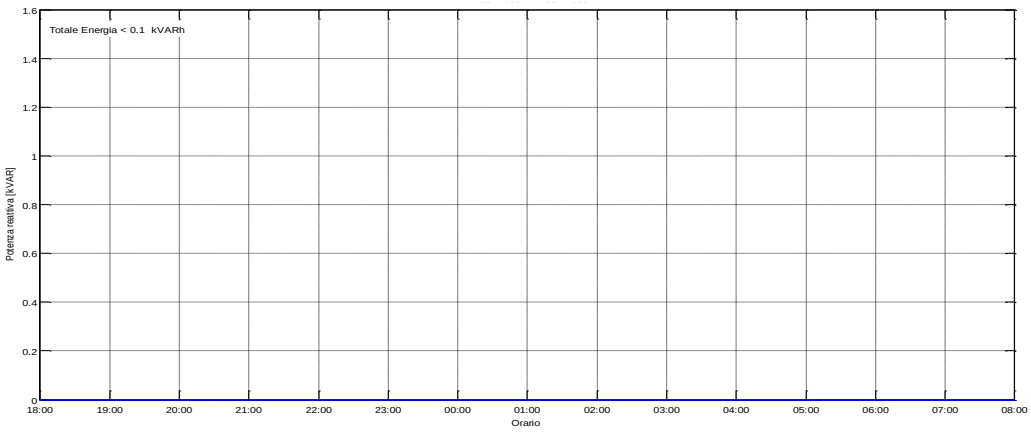


Figura 83: Potenza Reattiva misurata il 19/07/2015 dalle ore 18:00

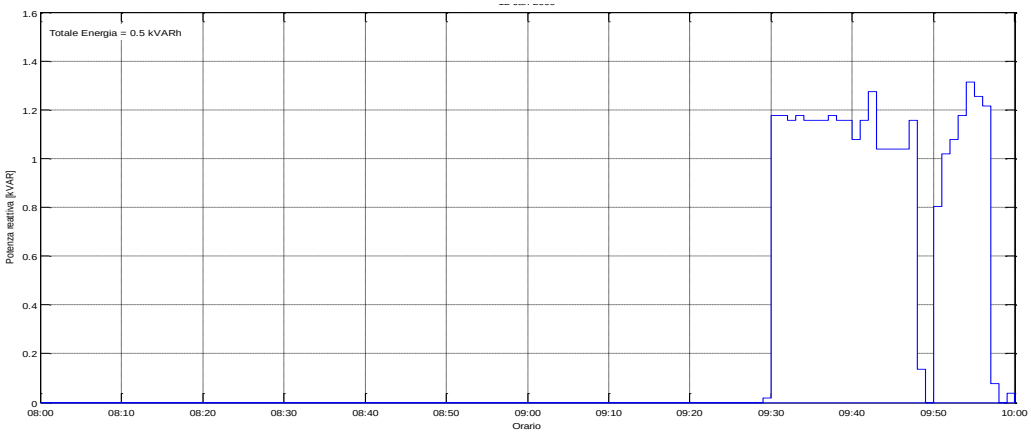


Figura 84: Potenza Reattiva misurata il 20/07/2015 dalle ore 08:00

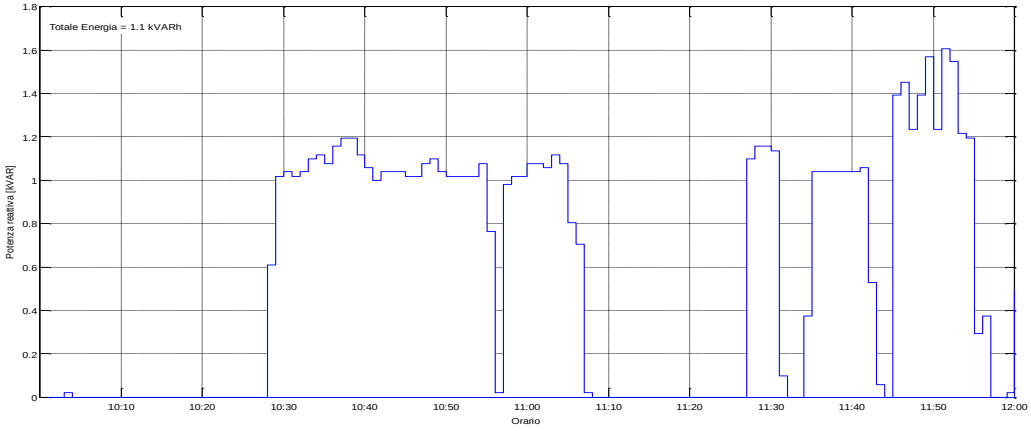


Figura 85: Potenza Reattiva misurata il 20/07/2015 dalle ore 10:00

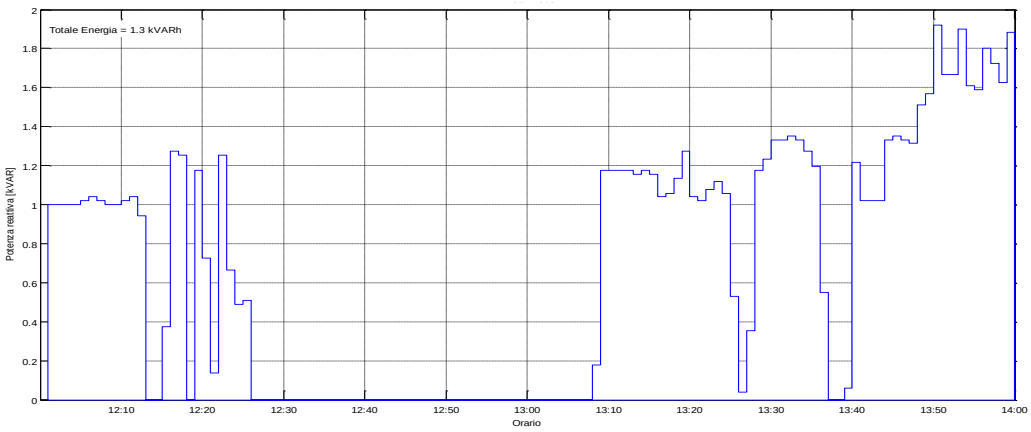


Figura 86: Potenza Reattiva misurata il 20/07/2015 dalle ore 12:00

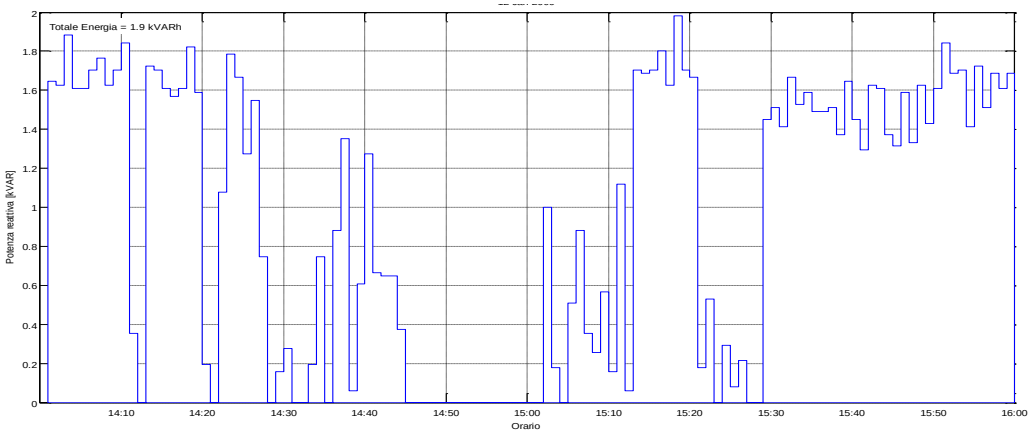


Figura 87: Potenza Reattiva misurata il 20/07/2015 dalle ore 14:00

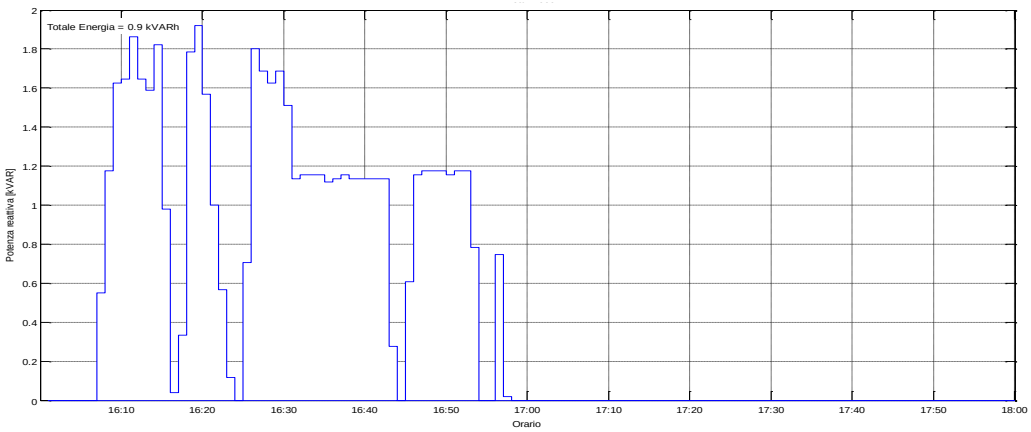


Figura 88: Potenza Reattiva misurata il 20/07/2015 dalle ore 16:00

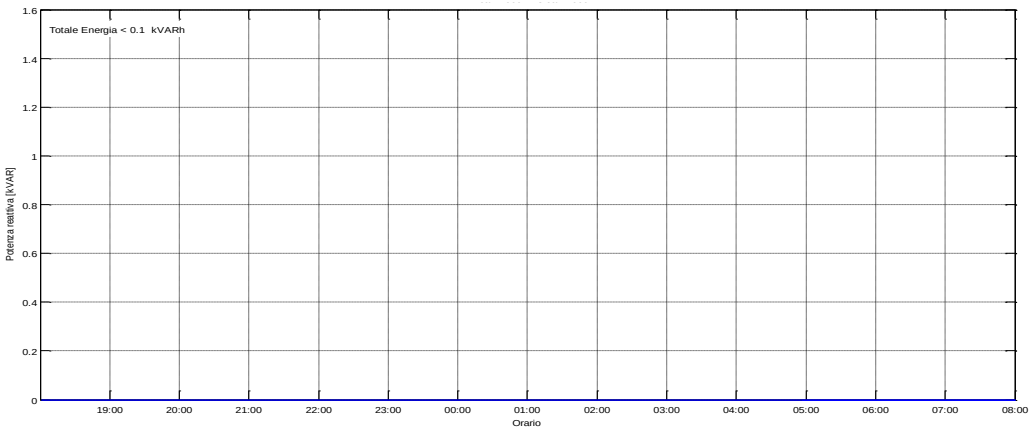


Figura 89: Potenza Reattiva misurata il 20/07/2015 dalle ore 19:00

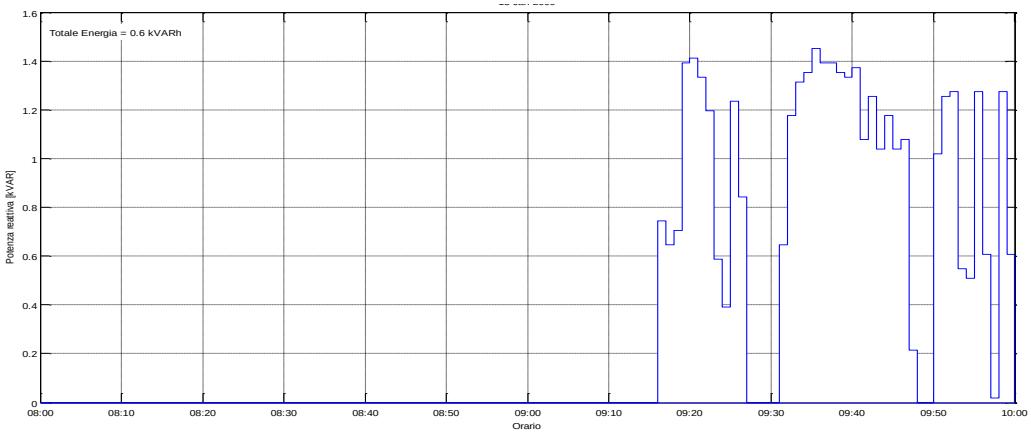


Figura 90: Potenza Reattiva misurata il 21/07/2015 dalle ore 08:00

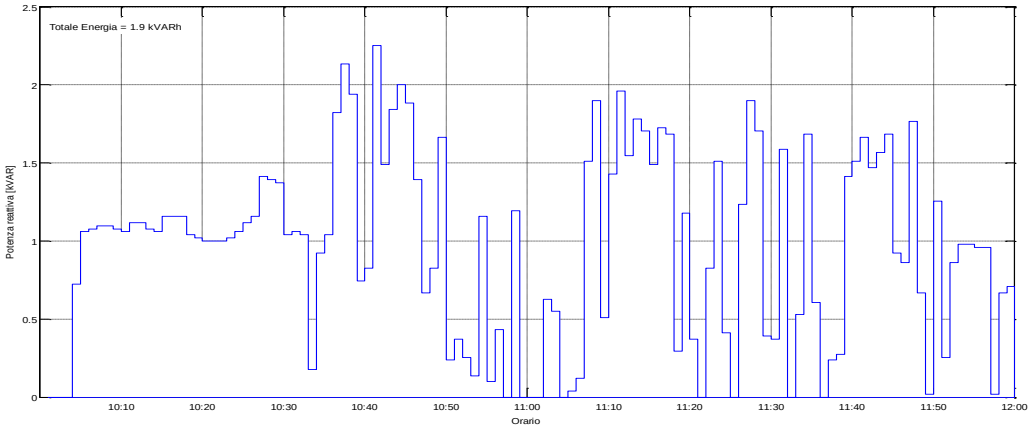


Figura 91: Potenza Reattiva misurata il 21/07/2015 dalle ore 10:00

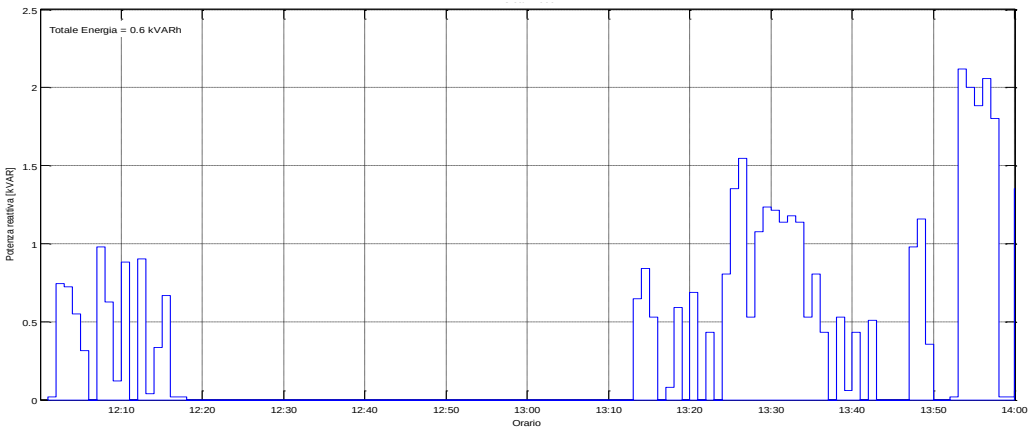


Figura 92: Potenza Reattiva misurata il 21/07/2015 dalle ore 12:00

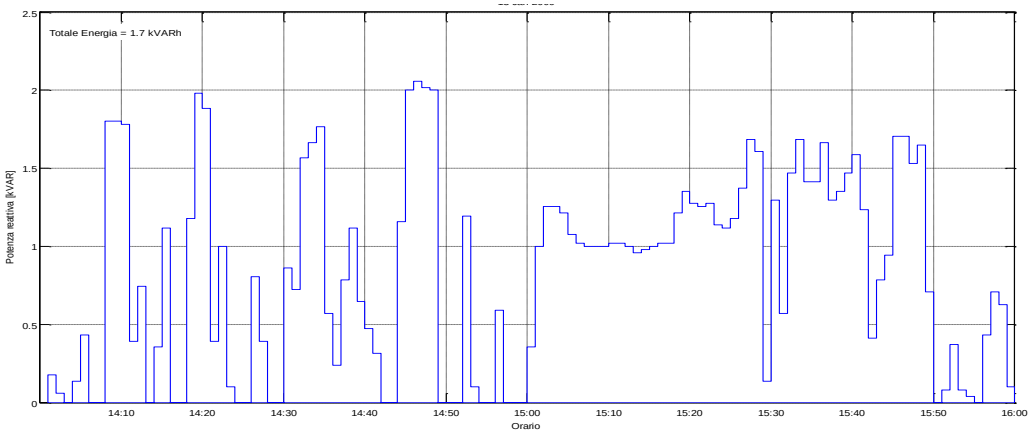


Figura 93: Potenza Reattiva misurata il 21/07/2015 dalle ore 14:00

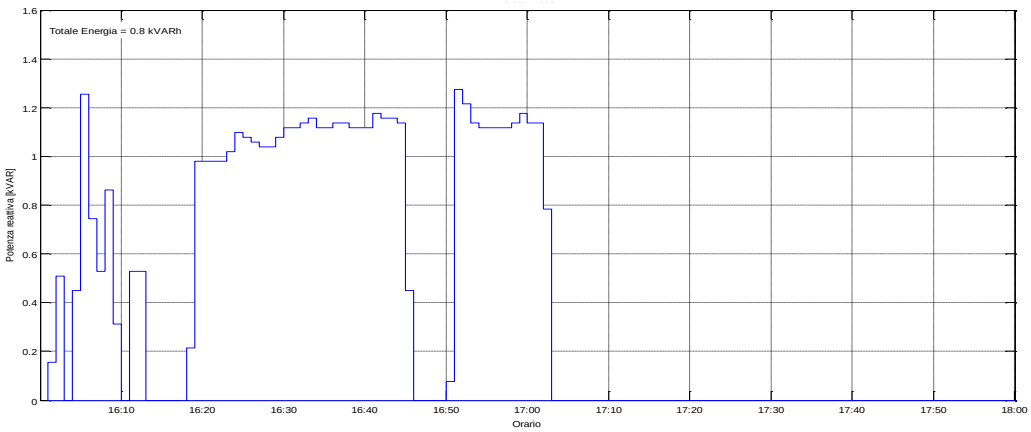


Figura 94: Potenza Reattiva misurata il 21/07/2015 dalle ore 16:00

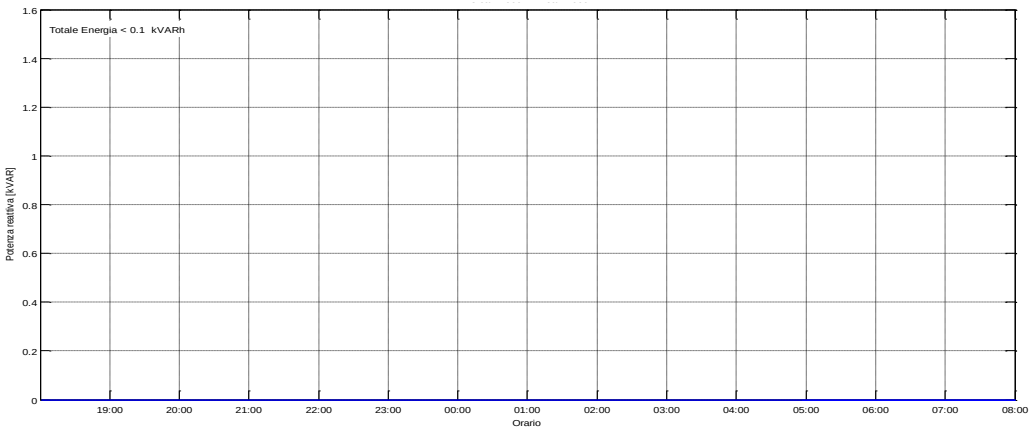


Figura 95: Potenza Reattiva misurata il 21/07/2015 dalle ore 19:00

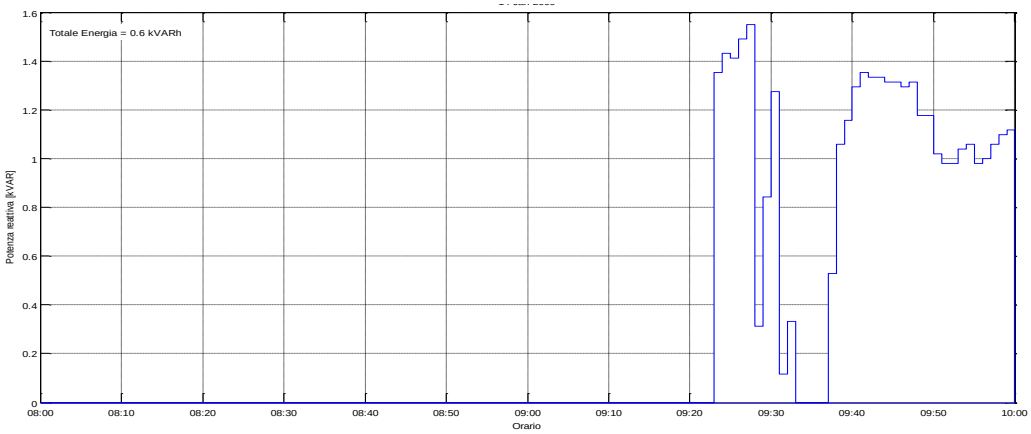


Figura 96: Potenza Reattiva misurata il 22/07/2015 dalle ore 08:00

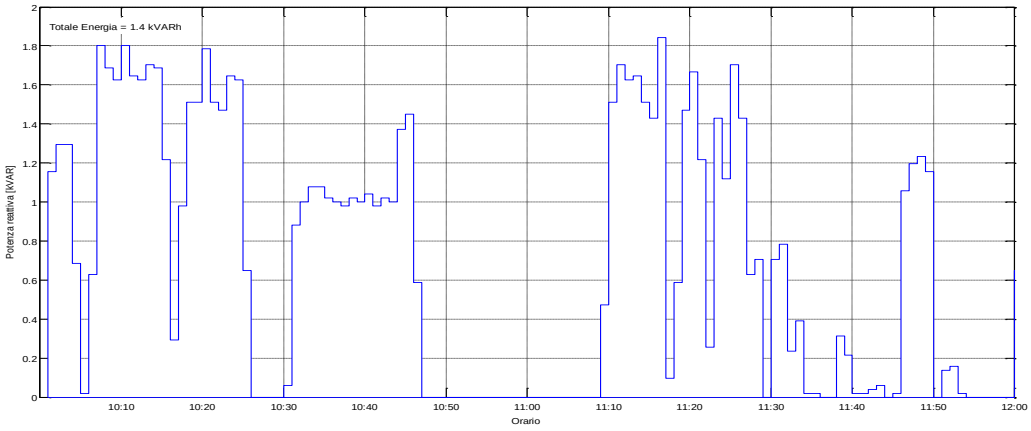


Figura 97: Potenza Reattiva misurata il 22/07/2015 dalle ore 10:00

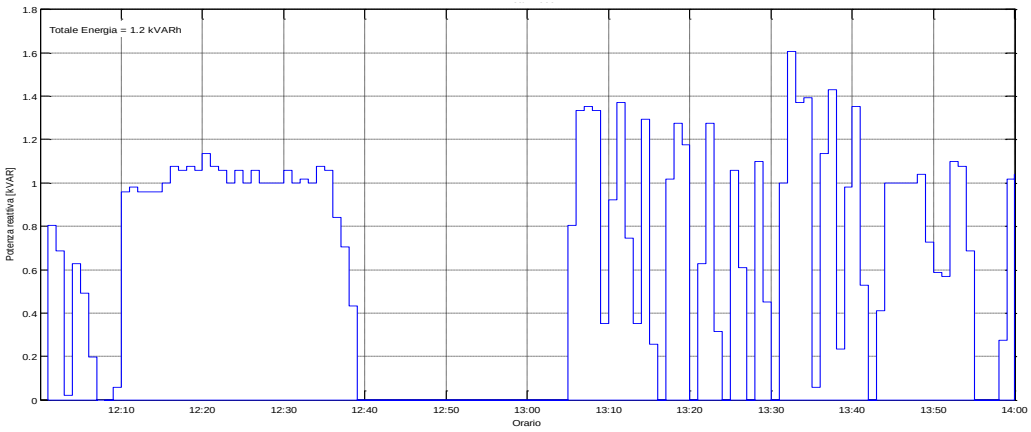


Figura 98: Potenza Reattiva misurata il 22/07/2015 dalle ore 12:00

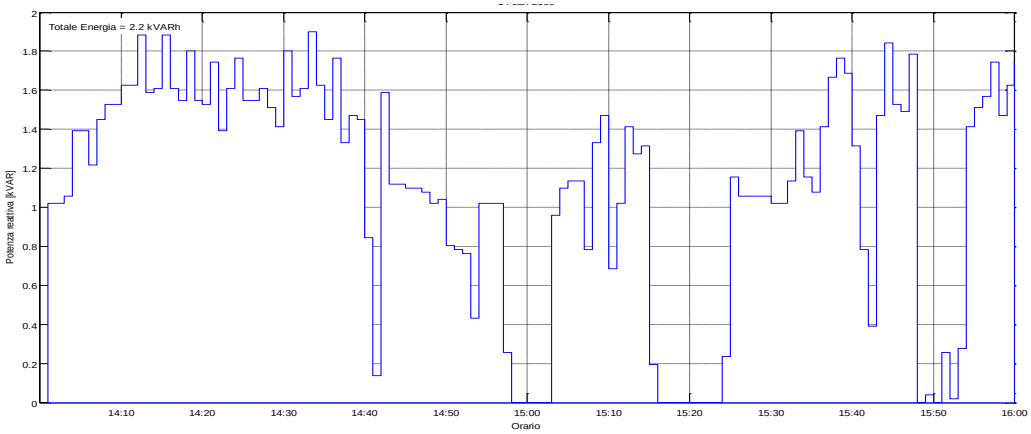


Figura 99: Potenza Reattiva misurata il 22/07/2015 dalle ore 14:00

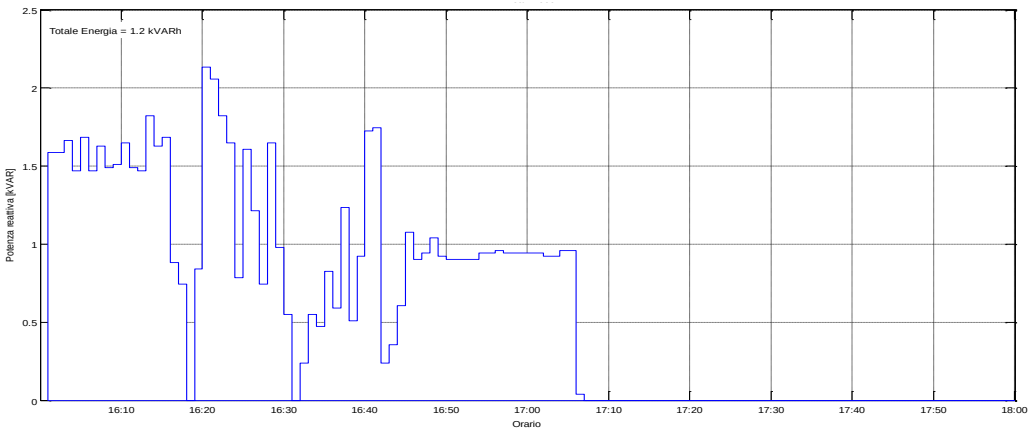


Figura 100: Potenza Reattiva misurata il 22/07/2015 dalle ore 16:00

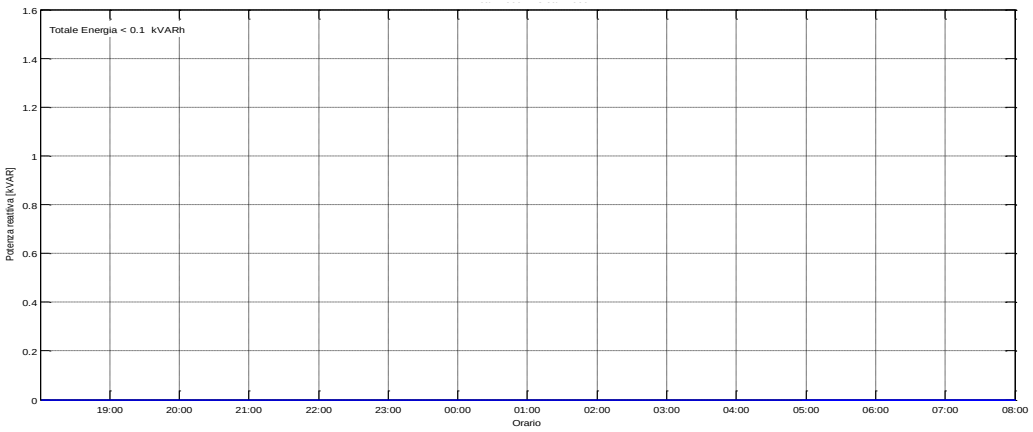


Figura 101: Potenza Reattiva misurata il 22/07/2015 dalle ore 19:00

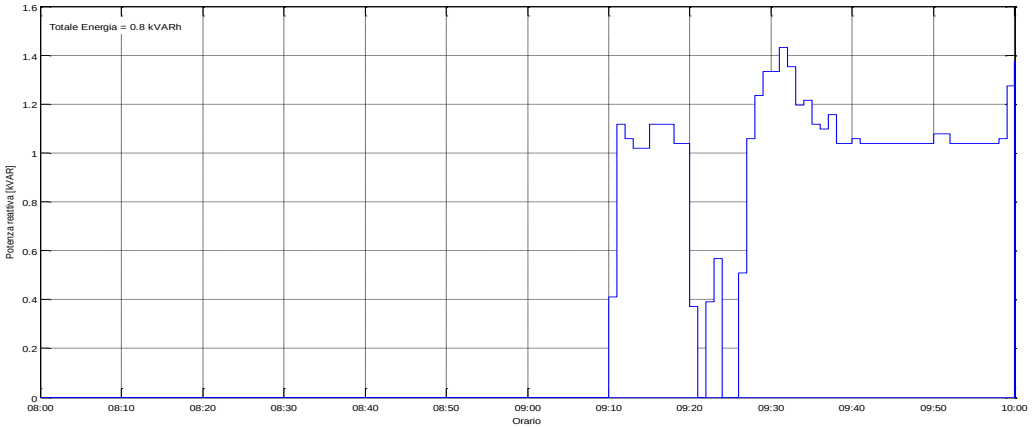


Figura 102: Potenza Reattiva misurata il 23/07/2015 dalle ore 08:00

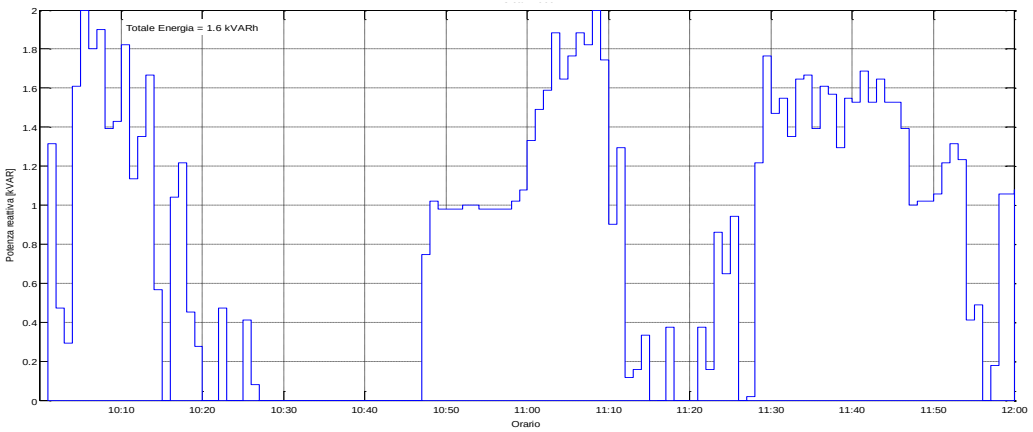


Figura 103: Potenza Reattiva misurata il 23/07/2015 dalle ore 10:00

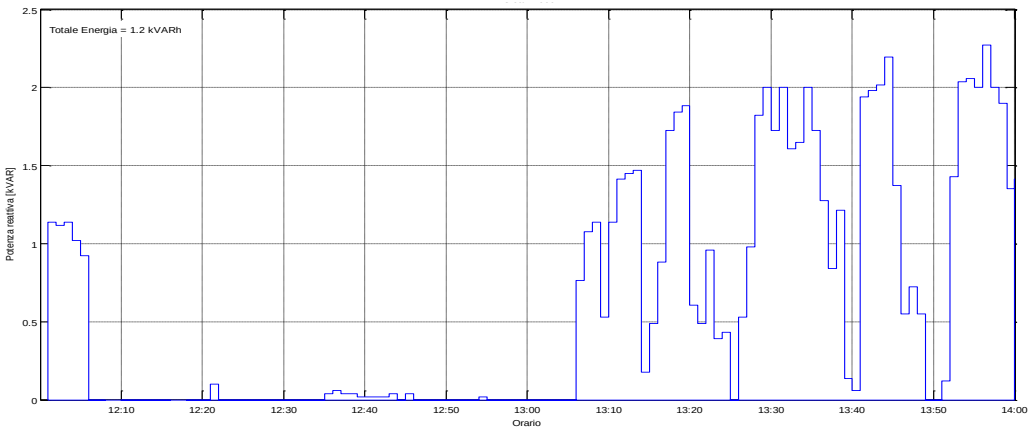


Figura 104: Potenza Reattiva misurata il 23/07/2015 dalle ore 12:00

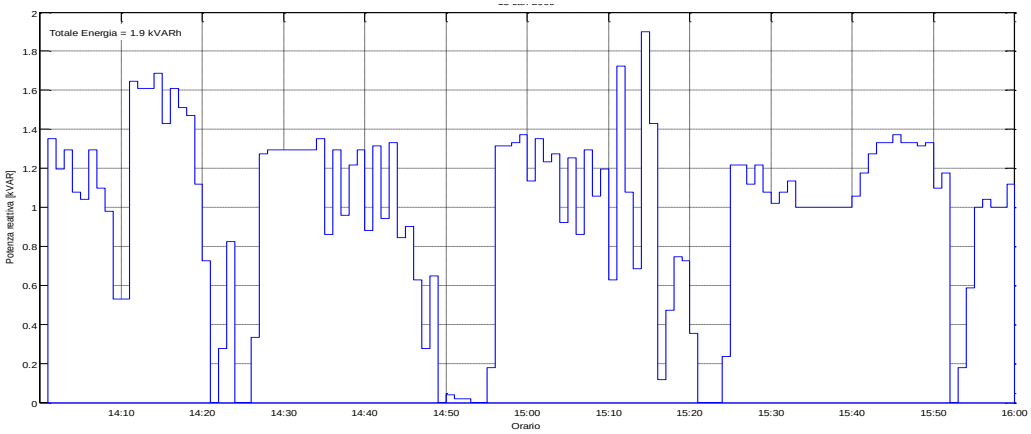


Figura 105: Potenza Reattiva misurata il 23/07/2015 dalle ore 14:00

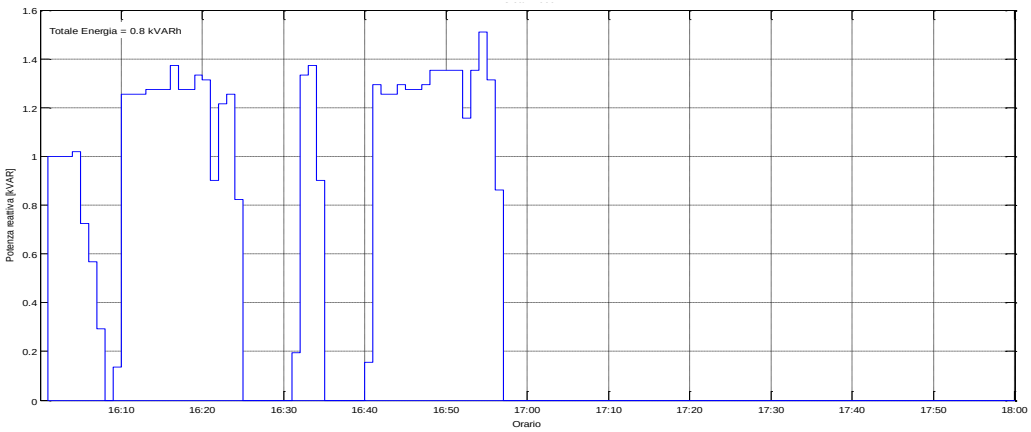


Figura 106: Potenza Reattiva misurata il 23/07/2015 dalle ore 16:00

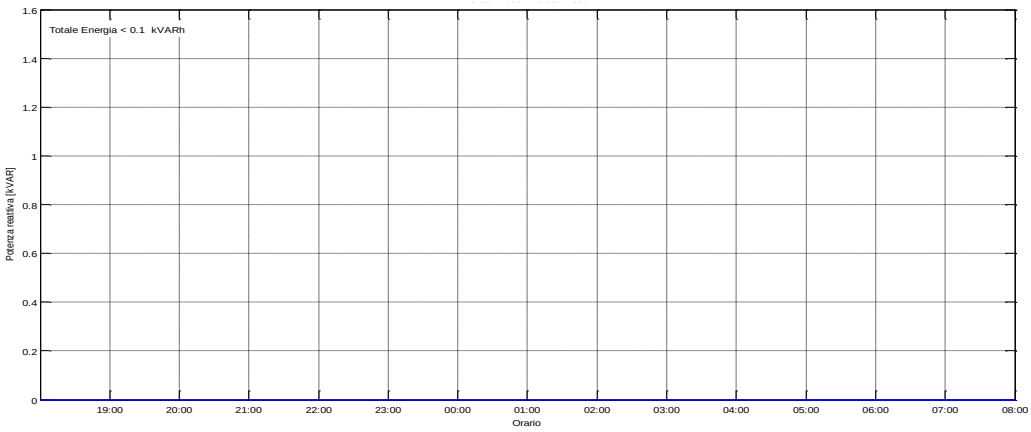


Figura 107: Potenza Reattiva misurata il 23/07/2015 dalle ore 19:00

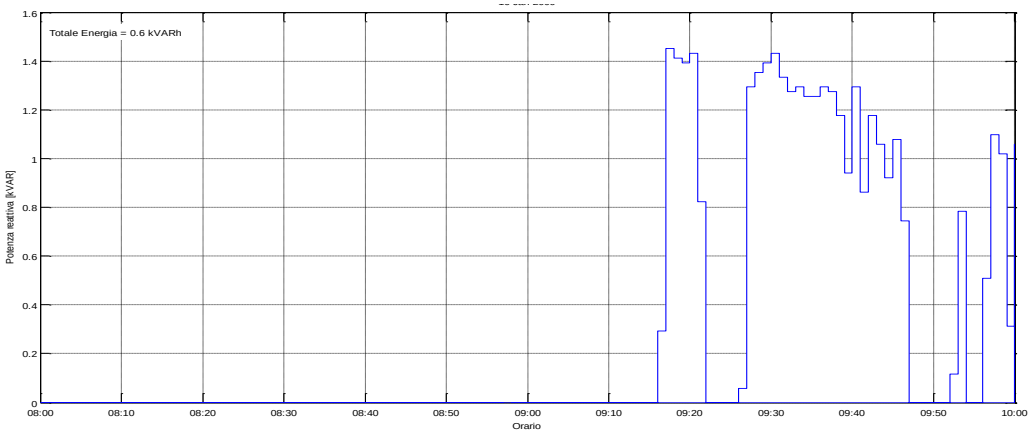


Figura 108: Potenza Reattiva misurata il 24/07/2015 dalle ore 08:00

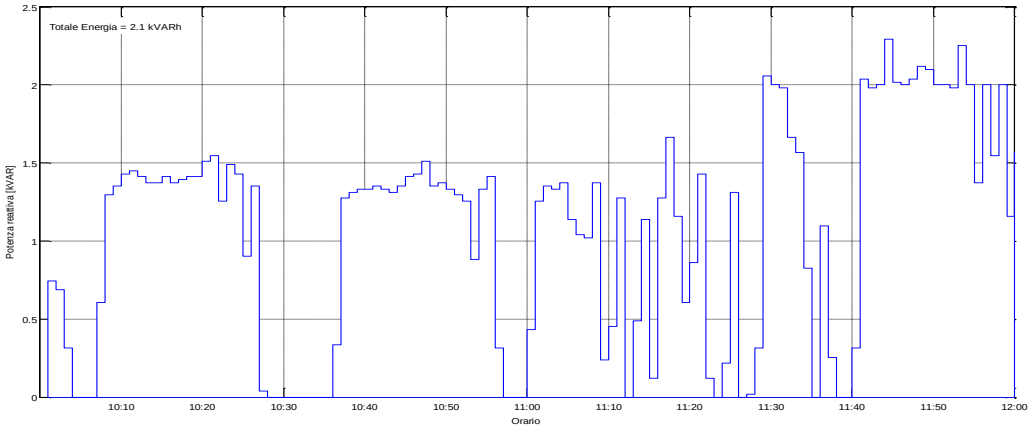


Figura 109: Potenza Reattiva misurata il 24/07/2015 dalle ore 10:00

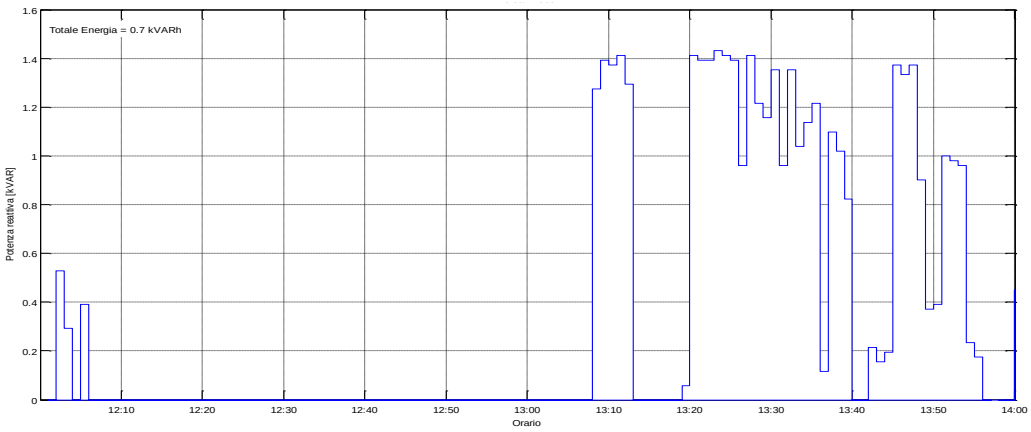


Figura 110: Potenza Reattiva misurata il 24/07/2015 dalle ore 12:00

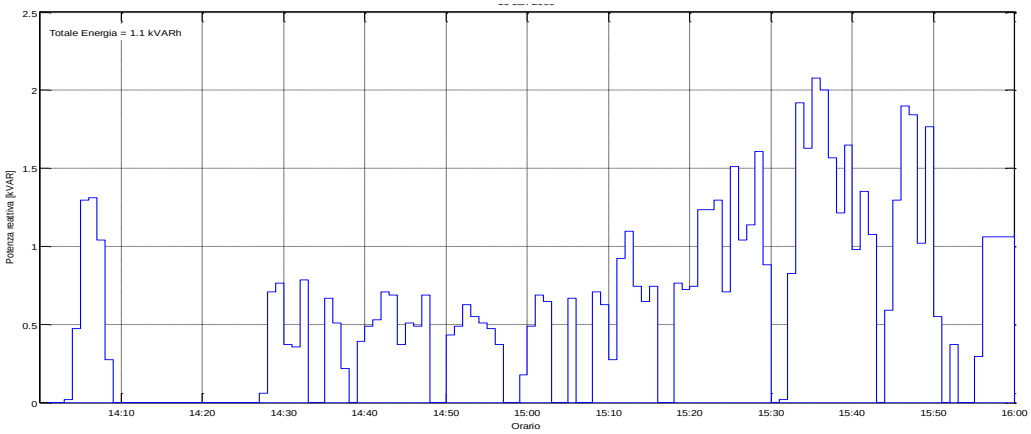


Figura 111: Potenza Reattiva misurata il 24/07/2015 dalle ore 14:00

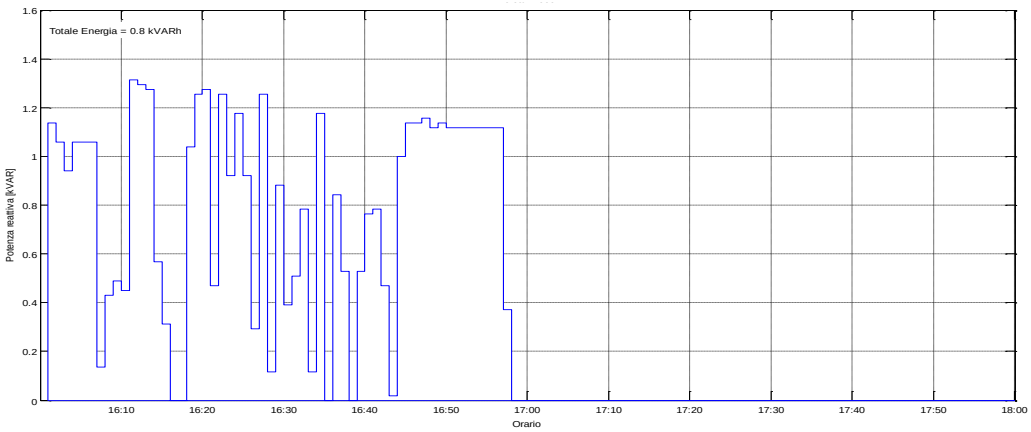


Figura 112: Potenza Reattiva misurata il 24/07/2015 dalle ore 16:00

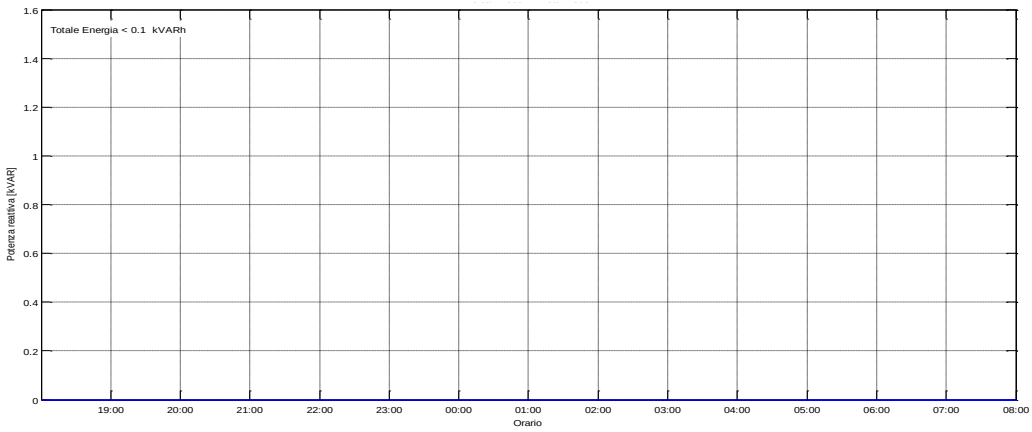


Figura 113: Potenza Reattiva misurata il 24/07/2015 dalle ore 19:00

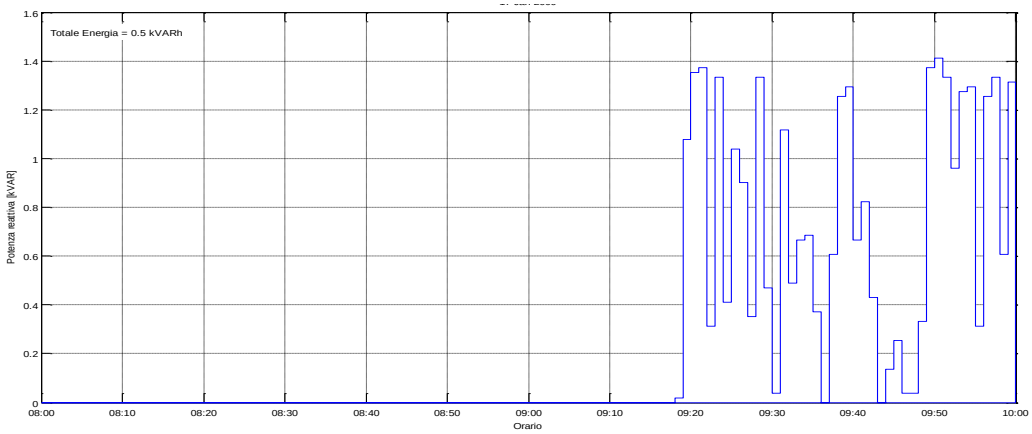


Figura 114: Potenza Reattiva misurata il 25/07/2015 dalle ore 08:00

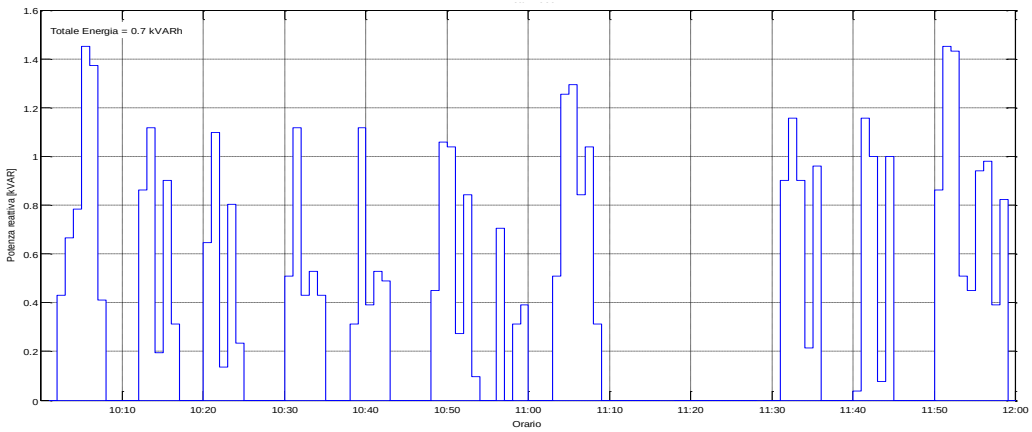


Figura 115: Potenza Reattiva misurata il 25/07/2015 dalle ore 10:00

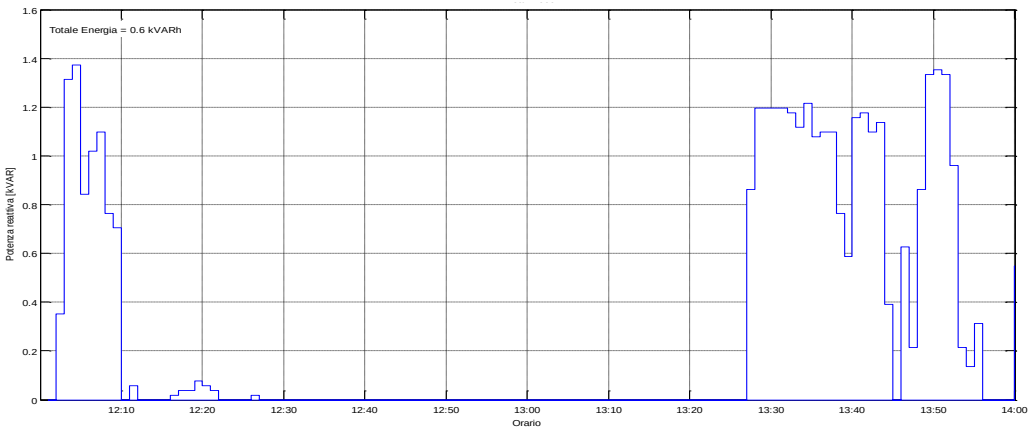


Figura 116: Potenza Reattiva misurata il 25/07/2015 dalle ore 12:00

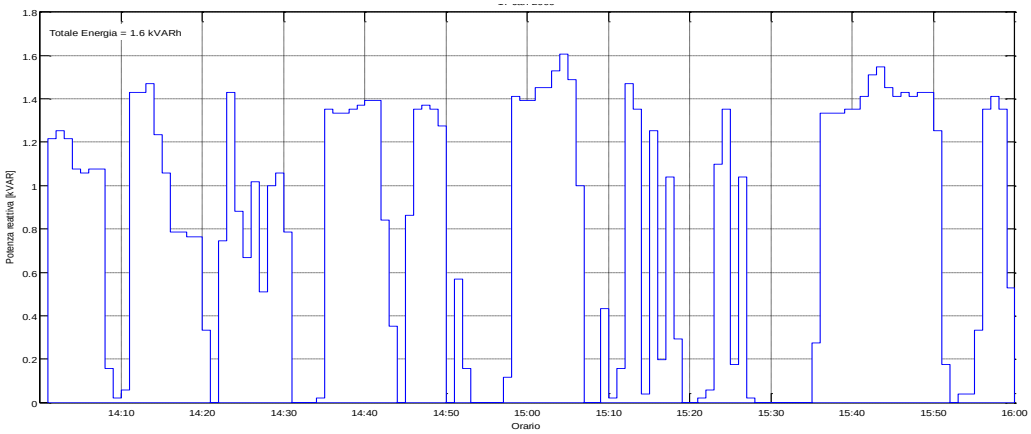


Figura 117: Potenza Reattiva misurata il 25/07/2015 dalle ore 14:00

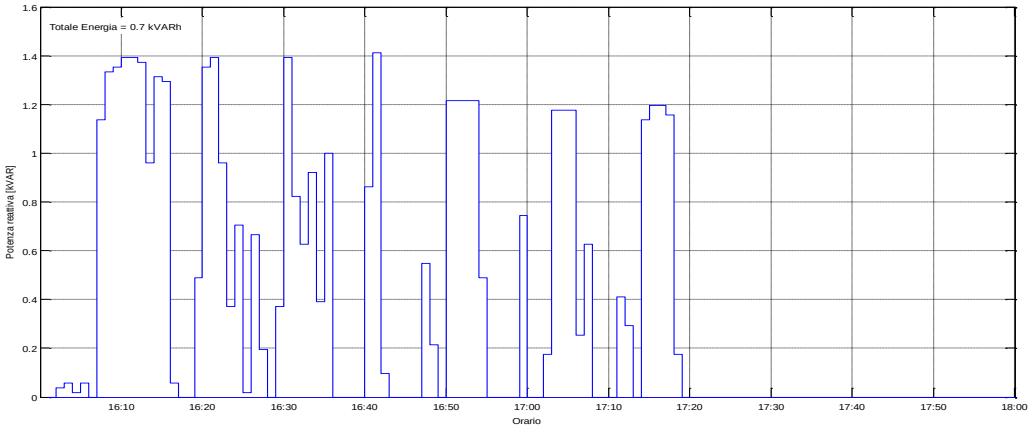


Figura 118: Potenza Reattiva misurata il 25/07/2015 dalle ore 16:00

4 Conclusioni

L’attività svolta ha portato alla realizzazione e caratterizzazione di un prototipo di rete con struttura ad albero e con funzionalità avanzate. Le prestazioni della rete si sono dimostrate molto interessanti sia per quanto riguarda le performance metrologiche, sia per quanto attiene gli aspetti di resilienza e sicurezza dei dati. Prove sperimentali condotte su una normale utenza elettrica hanno mostrato prestazioni in linea con dispositivi di misura di tipologia commerciale.

5 Riferimenti bibliografici

1. G. Bucci, E. Fiorucci, F. Ciancetta, M. Luiso, “Measuring System for Microelectric Power”, IEEE Transactions on Instrumentation and Measurement, vol. 63, p. 410-421, ISSN: 00189456, doi: 10.1109/TIM.2013.2280475, February 2014.
2. G. Aurilio, D. Gallo, G. Graditi, C. Landi, M. Luiso, “A low cost smart meter network for a smart utility”, Instrumentation and Measurement Technology Conference (I2MTC) Proceedings, 2014 IEEE International, 12-15 May 2014, Montevideo, Uruguay, Pages 380 - 385, DOI: 10.1109/I2MTC.2014.6860772, ISBN: 978-1-4673-6385-3.
3. G. Aurilio, D. Gallo, C. Landi, M. Luiso, V. Cigolotti, G. Graditi, “Low cost combined voltage and current transducer for Smart Meters”, Instrumentation and Measurement Technology Conference (I2MTC) Proceedings, 2014 IEEE International, 12-15 May 2014, Montevideo, Uruguay, Pages 1459 - 1464, DOI: 10.1109/I2MTC.2014.6860987, ISBN: 978-1-4673-6385-3
4. D. Gallo, C. Landi, M. Luiso, “Power Meter Verification Issue: Reactive Power Measurement in Non Sinusoidal Conditions”, Instrumentation and Measurement Technology Conference (I2MTC) Proceedings, 2015 IEEE International, 11-14 May 2015, Pisa, Italy, Pages 1255-1260, ISBN 978-1-4799-6144-6
5. G. Bucci, E. Fiorucci, F. Ciancetta, D. Gallo, C. Landi, M. Luiso, “Embedded Power and Energy Measurement System Based on an Analog Multiplier”, Instrumentation and Measurement, IEEE Transactions on, ISSN: 00189456, Volume: 62, Issue: 8 DOI: 10.1109/TIM.2013.2255989, Publication Year: 2013, Page(s): 2248 – 2257.
6. F. Ciancetta, E. Fiorucci, D. Gallo, C. Landi, M. Luiso, “A Web Service Interface for a Distributed Measurement System Based on Decentralized Sharing Network”, Sensors & Transducers, Vol. 153, No. 6, June 2013, pp. 209-218, ISSN 2306-8515, ISSN 1726-5479.
7. G. Del Prete, D. Gallo, C. Landi, M. Luiso, “Real-Time Smart Meters Network For Energy Management”, ACTA IMEKO, ISSN: 2221-870X, Vol 2, No 1 (2013), pp.40-48.
8. D. Gallo, C. Landi, M. Luiso, R. Morello, “Optimization of Experimental Model Parameter Identification for Energy Storage Systems”, Energies 2013, 6(9), 4572-4590; doi:10.3390/en6094572, ISSN: 1996-1073.
9. G. Bucci, E. Fiorucci, F. Ciancetta, M. Luiso, “A Measuring System for Micro Electric Power”, Instrumentation and Measurement, IEEE Transactions on, ISSN: 00189456, in press.
10. D. Gallo, C. Landi, M. Luiso, E. Fiorucci, “Survey on Voltage Dip Measurement in Standard Framework”, Instrumentation and Measurement, IEEE Transactions on, ISSN: 00189456, in press.

11. G. Aurilio, G. Crotti, D. Gallo, D. Giordano, C. Landi, M. Luiso, “MV divider with fiber optic insulation”, IEEE International Workshop on Applied Measurements for Power Systems, September 25-27 2013, Aachen, Germany.
12. D. Gallo, C. Landi, M. Luiso, A. Rosano, “Experimental Validation of Mathematical Models of Storage Systems for Smart Grids”, IEEE International Workshop on Applied Measurements for Power Systems, September 25-27 2013, Aachen, Germany.
13. D. Gallo, C. Landi, M. Luiso, and E. Fiorucci, “Analysis of a photovoltaic system: AC and DC power quality,” WSEAS Transactions on Power Systems, ISSN, vol. 5060, pp. 45–55, 1790.

6 Curriculum scientifico del gruppo di lavoro impegnato nell'attività

Il gruppo di lavoro è costituito da eccellenze nell'ambito tecnico-scientifico di riferimento. Tra le diverse attività e qualifiche dei componenti del gruppo stesso, si può evidenziare che:

- sono membri del comitato scientifico della collana editoriale Automazione delle Misure e Controllo edita da Franco Angeli.
- sono revisori di progetti di ricerca di rilevante interesse nazionale: nel 2002 sono stati inseriti nell'albo degli esperti del MIUR per la valutazione di progetti di ricerca e sviluppo precompetitivo.
- è ricoperta la carica di Presidenza del CT 85/66 - Strumentazione di misura, di controllo e da laboratorio del CEI (Comitato Elettrotecnico Italiano).
- è ricoperta la carica di Presidenza del Consorzio Interuniversitario ricerca Me.S.E. - Metriche e tecnologie di misura sui Sistemi Elettrici.
- sono ricoperte le carica di Responsabile Scientifico di importanti progetti di ricerca e sviluppo industriale finanziati da MiSE e MIUR
- sono revisori delle più importanti riviste internazionali del settore Misure (IEEE Trans. on Instrumentation and Measurements, Measurement, Sensors and Interface, etc.).
- sono autori di più di trecento lavori scientifici pubblicati su riviste internazionali e nazionali ed in atti di congressi internazionali e nazionali.

Le principali tematiche di ricerca affrontate dal gruppo riguardano: (a) messa a punto di metodi di misura innovativi per la caratterizzazione, collaudo e diagnostica di componenti, apparecchiature e sistemi elettrici ed elettronici; (b) progettazione, realizzazione e sviluppo di strumenti di misura a microprocessori con architetture innovative, idonee a funzionare in tempo reale; (c) metodi, componenti e sistemi per la misura della potenza ed energia in regime distorto e la valutazione della qualità dell'alimentazione elettrica; (d) caratterizzazione di componenti e di impianti dell'energia elettrica da fonti rinnovabili, e) sviluppo di tecniche e dispositivi per la caratterizzazione e l'ottimizzazione delle prestazioni dei sistemi elettrici di potenza; f) realizzazione di trasduttori di tensione e corrente ad alta precisione e larga banda per la misura di potenza elettrica e della sua qualità.