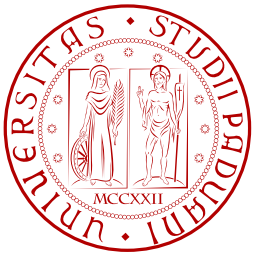

Università degli studi di Padova

FACOLTÀ DI INGEGNERIA ELETTRONICA ED INFORMATICA

CORSO DI LAUREA TRIENNALE IN INGEGNERIA ELETTRONICA

Sensore wireless per misure di potenza attiva in reti ZigBee/IEEE1451

Laureando: Marco Piovesan

Matricola: 540468 - IL

Relatore: Prof. Claudio Narduzzi

Padova, 21 Luglio 2010

Anno Accademico 2009/2010

...alla mia famiglia
e a Elena

Sommario

Questa tesi ha origine da un duplice scopo. Il primo obiettivo prevedeva lo studio di una demoboard di sviluppo Energy Meter di Microchip, che viene impiegata per misurare la potenza attiva assorbita da un carico alimentato a tensione di rete.
Il secondo obiettivo consisteva nell'utilizzo di un microcontrollore PIC di Microchip (TIM) per acquisire le informazioni provenienti dalla demoboard Energy Meter. Si è anche previsto che il TIM elaborasse l'informazione acquisita e la comunicasse attraverso un'interfaccia wireless ZigBee (TII) definita nello standard IEEE 802.15.4 a 2.4 GHz a un altro PIC Microchip (NCAP). L'NCAP organizza le informazioni ricevute da uno o più TIM rendendole disponibili in una Network principale (NI), come per esempio Internet.
Dall'unione di questi due obiettivi si è realizzato quello che prende il nome di *Smart Transducer* definito nello standard IEEE1451 per la realizzazione di reti di sensori.
Un tale dispositivo trova impiego sia nelle già note misurazioni dell'energia utilizzata da un'utenza passiva, sia nelle più attuali *Smart Grid*, cioè utenze che oltre a consumare energia possono anche immettere potenza attiva in linea per mezzo, ad esempio, di un sistema fotovoltaico.

Indice

Elenco delle tabelle

Elenco delle figure

Elenco dei listati

Capitolo 1

Standard IEEE 1451

1.1 Introduzione

La realizzazione di una rete di trasduttori secondo uno standard unico comporta molteplici vantaggi, tra i quali:

- la possibilità di definire una rete di trasduttori non vincolata a case produttrici;
- avere un modello comune per la gestione dei dati provenienti dai trasduttori, facilitando cosí le operazioni di controllo, configurazione e calibrazione dei trasduttori stessi;
- l'installazione, l'aggiornamento, la sostituzione o la rimozione dei trasduttori nella rete in modalità plug and play;
- la possibilità di definire per ciascun trasduttore un data sheet elettronico contenente tutte le informazioni necessarie per gestire il trasduttore stesso;
- l'accesso ai trasduttori sia in modalità wired che wireless, attraverso una vasta scelta di mezzi fisici.

Queste considerazioni hanno dato vita alla famiglia di standard IEEE 1451, un insieme di documenti elaborati negli ultimi dieci anni dall'IEEE (Institute of Electric and Electronic Engineers), che prevede la definizione di un'interfaccia standard per le reti di trasduttori intelligenti. Questa famiglia di standard definisce un'architettura di base della rete che consente di modificarne la configurazione in modalità plug and play e stabilisce le modalità d'accesso a una rete di trasduttori differenti per mezzo di comandi e procedure standardizzati. Inoltre è prevista l'aggiunta di data sheets elettronici, i TEDS (Transducer Elestronic Data Sheet), che contengono le informazioni relative al trasduttore stesso,

al costruttore, al range di misura, all'accuratezza, ai dati di taratura e molto altro. La direttiva proposta considera un modello di dispositivo programmabile e indipendente dalla rete a cui è collegato (1451.1), dotato di un'interfaccia digitale (1451.2) e di un protocollo di comunicazione che consente di accedere al trasduttore attraverso il microprocessore.

1.2 Smart transducer

In questo capitolo viene fornita una visione introduttiva generale sui cosiddetti smart transducers - ovvero trasduttori intelligenti - sulle loro applicazioni, sul loro inserimento in una rete e sulla necessità di uno standard che ne uniformi le caratteristiche di comunicazione.
Che cos'è un trasduttore intelligente?
Innanzitutto un trasduttore è un dispositivo che converte l'energia da una forma a un'altra. Nell'ambito degli strumenti di misurazione, un trasduttore può essere chiamato sensore oppure attuatore:

- un sensore è un trasduttore che genera un segnale elettrico proporzionale a un segnale fisico, biologico o chimico;

- un attuatore è un trasduttore che intraprende un'azione fisica in risposta a un segnale elettrico.

Uno smart transducer è l'integrazione di un sensore o un attuatore, analogico oppure digitale, con un'unità di elaborazione e un'interfaccia di comunicazione. Esso costituisce una piccola unità che integra, in generale, un trasduttore, un microprocessore e un modulo di comunicazione, oltre al software necessario alla taratura, al condizionamento del segnale, all'auto-diagnostica e ai protocolli di comunicazione.

La sua struttura è quindi quella presentata in figura 1.1 (a), in cui si possono distinguere i quattro blocchi fondamentali: l'output (analogico) del sensore viene condizionato e amplificato, quindi convertito in forma numerica. Il segnale numerico può essere elaborato dall'applicazione di misura (o di controllo) residente sul microprocessore e quindi inviato a un sistema remoto attraverso una connessione di rete. Considerando il funzionamento inverso, un host remoto può inviare comandi via rete per comandare l'attuatore.

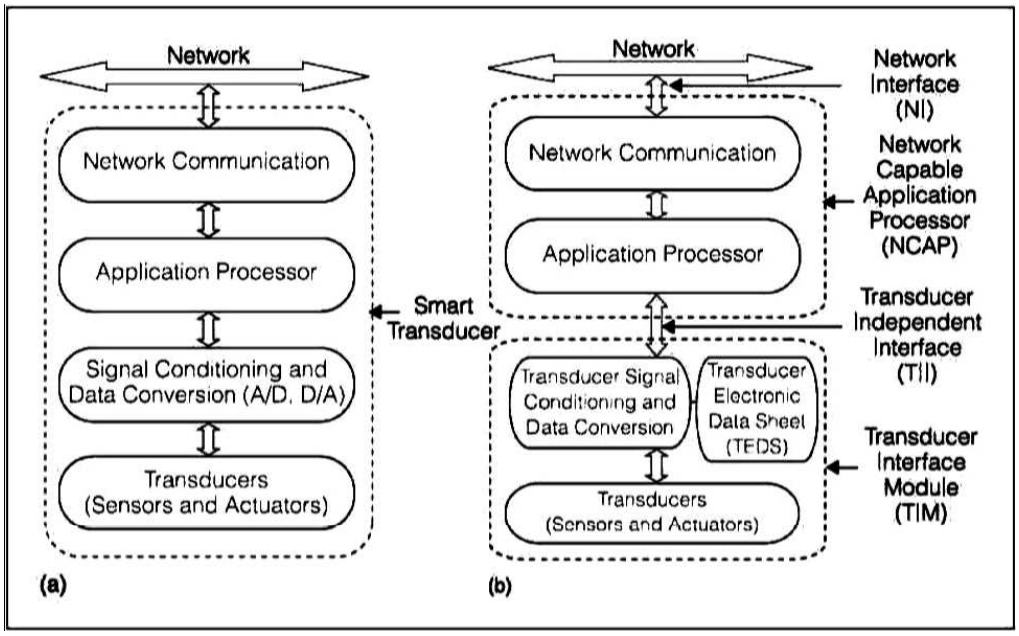

Figura 1.1: (a) Schema a blocchi di uno smart transducer; (b) Smart transducer con TEDS suddiviso in NCAP, TIM con TII

1.3 Smart transducers IEEE 1451

Uno smart transducer ha la sola capacità di generare una corretta rappresentazione di una quantità monitorata o controllata. Tuttavia la sempre maggiore pervasività dei trasduttori induce l'industria tecnologica a cercare di realizzare un'interfaccia standardizzata che semplifichi l'integrazione di tali dispositivi e la loro intercomunicazione, all'interno di reti più o meno estese che possono comprendere anche unità provenienti da produttori diversi. In risposta a questo tipo di esigenza, IEEE ha pensato alla definizione di uno standard, o meglio di una famiglia di standard per l'interfaccia dei trasduttori intelligenti, mirante a semplificare l'inserimento di tali dispositivi in un contesto di rete. Le funzionalità aggiuntive di uno smart transducer IEEE 1451 sono, ad esempio, la capacità di autoidentificazione, autodescrizione, autodiagnostica, autocalibrazione, cognizione di tempo e localizzazione, elaborazione dei dati, notifica di errori, e l'impiego di un formato standard per i dati e per il protocollo (o i protocolli) di comunicazione.
In seguito all'introduzione dello standard IEEE 1451 il modello dello smart transducer riportato in figura 1.1a viene modificato, ottenendo il modello di figura 1.1 (b), nel quale si possono vedere due variazioni principali: il suo partizionamento in due unità e l'intro-

duzione del Transducer Electronic Data Sheet (TEDS).
Le due unità che compongono lo smart transducer sono denominate:
TIM - Transducer Interface Module e NCAP - Network Capable Application Processor.
La prima contiene i trasduttori, ovvero sensori e/o attuatori, anche in combinazione mista (fino a un massimo di 255), oltre all'hardware di acquisizione dati e di condizionamento del segnale, mentre la seconda unità è di fatto un nodo della rete che attua le funzioni a livello applicativo e di comunicazione.
TIM e NCAP comunicano tra loro tramite un'interfaccia indipendente dal tipo di sensore/attuatore che viene denominata TII (Transducer Independent Interface).
Oltre alle specifiche sull'interfaccia di comunicazione lo standard dà informazioni relative al formato dei TEDS (Transducer Electronic Data Sheets).
Un TEDS è una sorta di cartellino di identificazione per un certo trasduttore. Esso riporta una certa quantità di informazioni sul sensore/attuatore, come per esempio l'identificazione del produttore, l'intervallo di misura, l'accuratezza e i dati riguardanti la taratura. Il TEDS può essere conservato in un'area di memoria di sola lettura, oppure si può scegliere di mantenere nella memoria RAM del TIM le parti di esso soggette a modifiche durante il normale funzionamento. Nello schema di figura 1.2 si riportano i vari tipi di TEDS previsti dallo standard. Solo quattro di essi sono obbligatori, mentre gli altri sono a discrezione del costruttore.

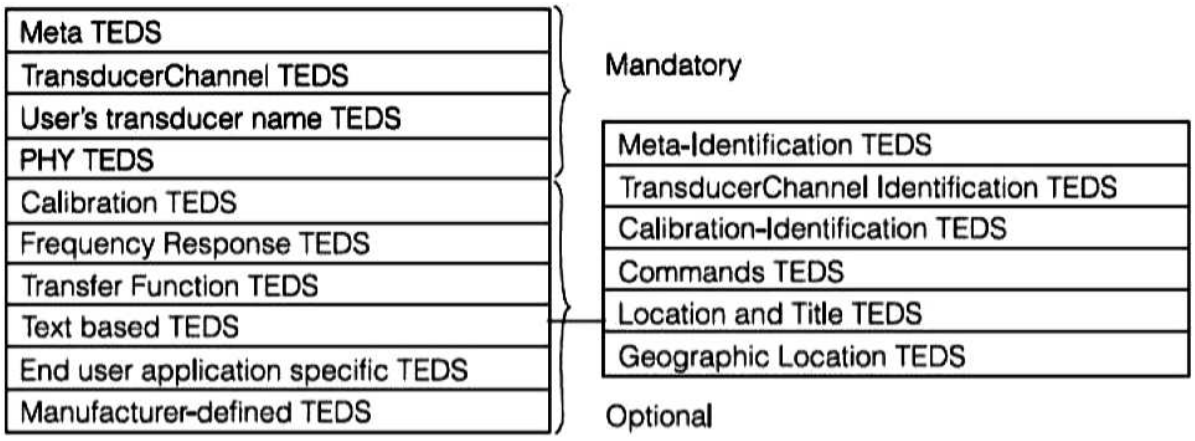

Figura 1.2: IEEE 1451.0 TEDS

1.4 La famiglia degli standard IEEE 1451

La famiglia di standard IEEE 1451 fornisce quindi un insieme di protocolli per applicazioni di misurazione e controllo distribuite, connesse tramite una rete cablata o wireless.

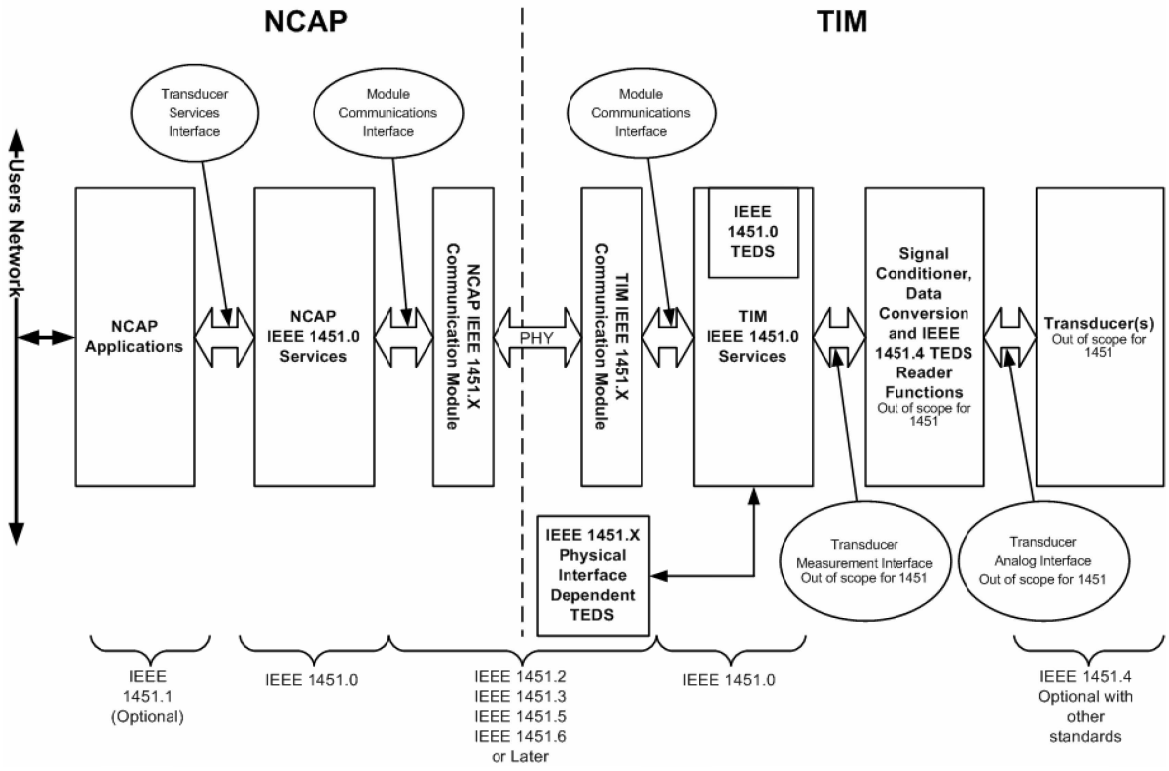

Figura 1.3: IEEE 1451 - Modello di riferimento

La figura 1.3 mostra l'architettura degli standard IEEE 1451. Lo standard IEEE 1451.0 definisce un insieme comune di comandi per accedere a sensori e attuatori fisicamente connessi in vari modi, come per esempio point-to-point, multi-drop o wireless. Per accedere a un TIM attraverso una rete esistono tre possibilità:

- IEEE 1451.1;
- IEEE 1451.0 Hyper Text Transfer Protocol (HTTP);
- Smart Transducer Web Service.

Inoltre i tipi di interfaccia fisica con cui il TIM si connette all'NCAP sono:

- point-to-point, definita in IEEE 1451.2-1997;
- multi-drop distribuito, definita in IEEE 1451.3-2003;

- wireless (WiFi, Bluetooth e ZigBee), definita in IEEE 1451.5-2007;
- CANopen, definita in IEEE p1451.6 (attualmente solo proposto);
- Radio Frequency Identification (RFID), definita in IEEE p1451.7 (anch'esso solo proposto).

L'obiettivo principale di IEEE 1451 è quello di permettere l'accesso a dati in formato standardizzato provenienti dai trasduttori, siano essi connessi direttamente al sistema oppure inseriti in una rete cablata o wireless.

1.5 Gli elementi di IEEE 1451

Vediamo brevemente quali sono gli scopi e le funzionalità dei vari elementi della famiglia IEEE 1451.

IEEE 1451.0

Il layer IEEE 1451.0 definisce un insieme di istruzioni e di comandi che sono comuni a tutti gli elementi di IEEE 1451 e indipendenti dal mezzo di comunicazione (1451.X) tra il trasduttore e l'NCAP. Tra questi ci sono le istruzioni di base per leggere e scrivere i dati del trasduttore, leggere e scrivere i TEDS e per inviare i comandi di configurazione e controllo al TIM: ciò rende semplice l'aggiunta di ulteriori layer fisici. L'obiettivo principale di IEEE 1451.0 è quello di ottenere un'interoperabilità a livello dati all'interno di reti di sensori connessi in varie modalità, cablate o wireless.

IEEE 1451.1

Il layer IEEE 1451.1 definisce un modello comune per i componenti di uno smart transducer inserito in una rete, oltre alle specifiche riguardanti l'interfaccia. L'architettura software di questo layer è definita attraverso i tre modelli seguenti:

- un modello dati specifica il tipo e la forma delle informazioni che vengono comunicate attraverso l'interfaccia tra gli oggetti definiti da IEEE 1451.1, nel caso di comunicazioni sia locali che remote;
- un modello oggetto specifica il tipo delle componenti software usate per progettare e implementare il sistema: in pratica tale modello fornisce una serie di blocchi con i quali costruire l'applicazione di controllo o misura;
- due modelli di comunicazione definiscono la sintassi delle interfacce software tra gli oggetti (o blocchi software) dell'applicazione e la rete. Di fatto questo layer si

concentra principalmente sulla comunicazione tra gli NCAP e tra un NCAP e altri nodi della rete.

IEEE 1451.2

Il layer IEEE 1451.2 definisce l'interfaccia di comunicazione e il formato dei TEDS nel caso di connessione point-to-point tra NCAP e trasduttore. Lo standard originale prevede l'utilizzo dell'interfaccia SPI (Serial Peripheral Interface) con l'aggiunta di alcune linee ulteriori per il controllo di flusso e per la temporizzazione. Esso è in corso di modifica per renderlo interfacciabile con IEEE 1451.0 e compatibile con altre interfacce comuni, tra cui UART.

IEEE 1451.3

Il layer IEEE 1451.3 definisce l'interfaccia di comunicazione e il formato dei TEDS nel caso di connessione multidrop tra NCAP e trasduttore. Esso permette di configurare i trasduttori di una rete come nodi che condividono la stessa linea di comunicazione.

IEEE 1451.4

Il layer IEEE 1451.4 definisce un'interfaccia mista per supportare trasduttori analogici che possano operare in modalità sia analogica che digitale. Lo scopo di questo layer è quello di aggiungere il TEDS ai vecchi sensori con interfaccia analogica. All'accensione del sistema il TEDS viene inviato tramite interfaccia digitale ad un unico filo (one-wire), dopodiché l'interfaccia viene commutata in modalità analogica in modo da poter essere utilizzata per trasportare l'informazione analogica dal trasduttore al sistema di misura.

IEEE 1451.5

Il layer IEEE 1451.5 definisce l'interfaccia di comunicazione e il formato dei TEDS nel caso di connessione wireless. Sono supportate diverse soluzioni wireless, come 802.11 (WiFi), 802.15.1 (Bluetooth) e 802.15.4 (ZigBee). La rete wireless tra i TIM e l'NCAP può anche essere eterogenea: l'NCAP può implementare diversi tipi di interfaccia senza fili per comunicare con i TIM. Ciascun TIM, invece, comunica con un solo NCAP e necessita quindi solamente dell'interfaccia appropriata.

IEEE p1451.6

Il layer IEEE p1451.6 (proposto) definisce l'interfaccia di comunicazione e il formato dei TEDS nel caso di connessione tramite CANopen.

IEEE p1451.7

Il layer IEEE p1451.7 (proposto) definisce l'interfaccia di comunicazione e il formato dei TEDS nel caso di connessione fra trasduttori e dispositivi RFID (Radio Frequency Identification).

1.6 Possibili benefici introdotti da IEEE 1451

I data-sheet elettronici (TEDS) introdotti da IEEE 1451 contengono, come già detto, un certo numero di informazioni riguardanti il produttore del trasduttore e il trasduttore stesso, in un formato standard. I principali benefici resi possibili dall'uso dei TEDS sono elencati di seguito.

- autoidentificazione: il sensore/trasduttore può identificare se stesso dinanzi al dispositivo host o, in generale, alla rete inviando le informazioni contenute nei propri TEDS;
- autodocumentazione: i TEDS possono essere aggiornati nel tempo e contenere informazioni utili nel lungo periodo, quali per esempio la localizzazione del sensore o uno storico degli interventi di manutenzione;
- il trasferimento automatico dei TEDS dal trasduttore alla rete riduce l'eventualità di errori umani, dal momento che non è più necessario inserire manualmente i dati in ogni dispositivo. Ciò naturalmente apporta una maggiore rapidità esecutiva;
- i TEDS permettono di facilitare l'installazione, l'aggiornamento e la manutenzione della rete di trasduttori, poiché grazie a essi si può raggiungere uno stato in cui ogni dispositivo si comporta in modo plug and play.

Un TIM e un NCAP basati entrambi su IEEE 1451 possono essere connessi agevolmente con un'interfaccia di comunicazione standardizzata e scambiare dati senza la necessità di alcuna modifica al software del sistema. Non servono driver aggiuntivi o altro per garantire il corretto funzionamento delle funzionalità di base del sistema e ciò è particolarmente vantaggioso per i produttori dei dispositivi, poiché possono per esempio scegliere di produrre solamente i TIM, senza entrare nel merito della loro interconnessione in rete, oppure realizzare un unico modulo che integri TIM e NCAP, adatto per essere inserito in una qualunque rete compatibile di trasduttori. In quest'ultimo caso l'interfaccia tra TIM e NCAP è nascosta e può essere realizzata in modo anche non conforme: il modulo infatti rimarrebbe comunque compatibile con IEEE 1451 a livello di rete.

1.7 Possibili applicazioni di IEEE 1451

Alcuni possibili campi di applicazione della tecnologia IEEE 1451 sono:

- **controllo e misurazione remoti:** il parametro misurato da un sensore situato su un TIM può essere letto a distanza attraverso l'NCAP, che può inviare il dato

su una rete locale o anche su Internet; inoltre qualsiasi stazione remota che abbia accesso alla rete può visualizzarne il valore. Una cosa analoga vale nel caso degli attuatori;

- **controllo e misurazione distribuiti:** quando un TIM è equipaggiato sia con sensori che con attuatori esso può realizzare misurazioni o funzioni di controllo, rispondendo agli ordini di un qualunque NCAP connesso alla stessa rete o a Internet;

- **controllo e misurazione collaborativi:** in questo caso due NCAP connessi rispettivamente a un TIM equipaggiato con un sensore e a uno equipaggiato con un attuatore possono effettuare procedure di misurazione e controllo in cooperazione.

1.8 IEEE 1451.0

Introduzione

Lo standard IEEE 1451.0 si posiziona, nello stack protocollare, tra l'applicazione e il protocollo di comunicazione scelto in base all'interfaccia fisica utilizzata, fungendo quindi a sua volta da convergence layer tra l'applicazione e la rete. Ovviamente anche questo protocollo riconosce i due dispositivi previsti dalla famiglia IEEE 1451 (NCAP e TIM) e si occupa di descriverne tutte le caratteristiche comuni, nonché le operazioni e le funzionalità che caratterizzano i dispositivi di una rete di sensori intelligenti, indipendentemente dal tipo di interfaccia fisica prescelta.

Conformità

Per ottenere la dicitura plug-and-play un dispositivo di questo tipo deve rispondere a determinate caratteristiche richieste a livello applicazione. Esse sono tutte specificate nella documentazione dello standard, ma in particolare si ricorda che sia gli NCAP che i TIM funzionanti secondo lo standard IEEE 1451.0 devono supportare un protocollo di comunicazione e un mezzo fisico definiti dalla stessa famiglia di standard IEEE 1451.

Rete

Lo standard 1451.0 non richiede alcun tipo specifico di rete fisica, pertanto la scelta è lasciata all'utente; tuttavia necessita che l'NCAP sia dotato del software e dell'hardware appropriato per la gestione della tipologia di rete prescelta.

Indirizzamento

Esistono due tipi di indirizzamento definiti da questo standard: il primo è di tipo fisico e se ne occupa appunto il physical layer, tramite il parametro destId (così riconosciuto dal Module Communication Interface); il secondo parametro per l'indirizzamento viene chiamato TransducerChannelNumber.

API

Lo standard definisce una Application Program Interface (API) per tutte le applicazioni che provvedono alla comunicazione tra la rete e il layer IEEE 1451 e tra il protocollo IEEE 1451.0 e i sottostanti layer fisici di comunicazione, di solito denominati in questo standard come layer IEEE 1451.X.
Lo standard 1451.0 definisce quindi due tipi di API: il primo è la Transducer Service interface, API del solo NCAP, utilizzata dalle applicazioni di misura e controllo lato utente, per accedere al layer IEEE 1451.0. Questa API contiene i metodi per leggere e scrivere i TEDS, per gestire i TransducerChannels e per inviare comandi di configurazione e controllo ai TIM. L'altra API, la Module Communication Interface, si colloca tra lo standard 1451.0 e un altro membro della famiglia 1451. Essa è un'interfaccia simmetrica che va implementata sia sull'NCAP che sul TIM e contiene metodi che dovrebbero essere implementati dal layer IEEE 1451.X e chiamati per iniziare le operazioni di comunicazione. Similarmente esistono metodi che devono essere implementati dal 1451.0 e invocati dal 1451.X per consegnare le informazioni inviate. Per mantenere una neutralità di linguaggio le funzioni e i parametri delle API sono descritti con il linguaggio Interface Definition Language (IDL).

1.9 IEEE 1451.5

Questo standard definisce un'interfaccia wireless per sensori e specifica i protocolli radio per tale interfaccia wireless. Inoltre definisce i moduli di comunicazione che connettono il Wireless Transducer Interface Module (WTIM) e il Network Capable Application Processor (NCAP), usando gli specifici protocolli radio e anche i TEDS per questi tipi di protocollo. Il WTIM è un dispositivo che comprende una Dot 5 Approved Radio (Dot5AR) - che nel caso di questa tesi è la tecnologia ZigBee -, il condizionamento del segnale, la conversione A/D (analogico/digitale) e uno o più trasduttori (sensori/attuatori). Poiché i WTIM possono avere interfacce Dot5AR diverse tra loro, l'NCAP dovrà avere almeno una Dot5AR per ogni tipo presente nei WTIM a cui deve essere associato.

Lo standard in questione è basato unicamente su interfacce wireless, ma non specifica le caratteristiche fisiche e tecniche né dei trasduttori né del sistema wireless. La nascita di questo protocollo è legata alla volontà di uniformare le specifiche e accomunare più tecnologie sotto un unico standard aperto, cercando di ridurre al minimo il rischio di incompatibilità e i costi di produzione. Le specifiche dell'IEEE 1451.5 riguardanti ZigBee indicano i requisiti che una rete di tale tipo deve avere affinché possa fungere da rete di trasporto per un sistema compatibile con IEEE 1451.

Convergence layer

Lo standard IEEE 1451.5 definisce il concetto di convergence layer, ossia un'interfaccia (in italiano, strato di convergenza) tra l'entità superiore (IEEE 1451.0) e la rete di trasporto. Esso infatti ha il compito di tradurre i comandi provenienti dal livello superiore in comandi specifici comprensibili al livello inferiore e viceversa. Nella documentazione del protocollo 1451.5 sono elencati solamente i metodi della Communication API tra 1451.5 e 1451.0 (appartenenti alla MCI del 1451.0). Per ciascuno di essi, assieme a una descrizione sintetica che ne spiega l'utilizzo in ambito 1451.5, vengono elencati nome, tipi di dato e parametri; questi metodi costituiscono il confine superiore del convergence layer. Il confine inferiore tra convergence layer e ZigBee nel nostro progetto è rappresentato dai metodi *read* e *write* presenti nell'interfaccia **P2PComm**.
Tale standard definisce i compiti specifici di NCAP e WTIM che sono:

- un NCAP può instradare dati verso sia una rete esterna, sia un trasduttore collegato a un WTIM;
- un NCAP può avere più WTIM associati;
- un NCAP può avere più interfacce radio (anche diverse);
- l'interfaccia ZigBee è gestita dal protocollo IEEE 1451.5 sia per gli NCAP che per i WTIM;
- un WTIM può associarsi a un solo NCAP;
- un WTIM può interfacciarsi a più trasduttori;
- è ammessa la comunicazione tra due WTIM.

Poiché nel nostro progetto ci occupiamo solo della comunicazione tra NCAP e un singolo WTIM, molti aspetti sopra elencati verranno tralasciati.

Diagrammi di stato dell'NCAP e del WTIM

Il funzionamento dell'interfaccia di comunicazione dell'NCAP è descrivibile attraverso una macchina a stati, dove a ogni stato corrispondono una o più azioni in base agli stimoli d'ingresso dell'NCAP stesso. Il funzionamento di tale macchina a stati è indicato in figura 1.4:

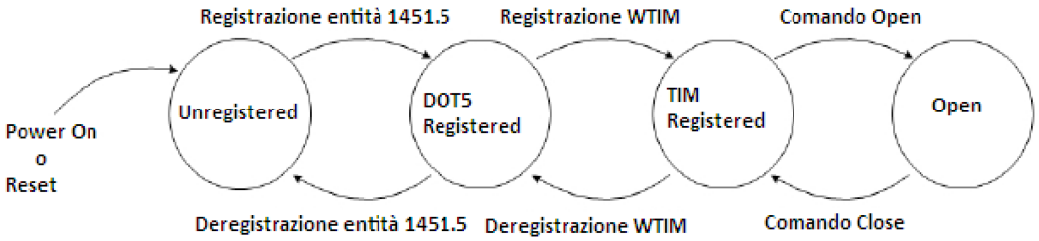

Figura 1.4: Macchina a stati NCAP

Lo stato iniziale di un NCAP dopo l'accensione o dopo il reset è UNREGISTERED. Dopo un processo di registrazione del modulo 1451.5, l'NCAP passa allo stato DOT5 REGISTERED per l'entità DOT5AR del protocollo 1451.5 e vi rimane finché non vi si associa un WTIM. L'NCAP ha il compito di mantenere separatamente lo stato di ognuna delle sue entità 1451.5, che a loro volta devono mantenere separatamente gli stati di tutti i loro WTIM associati. Quando un'entità 1451.5 registra uno o più WTIM, l'NCAP passa allo stato TIM REGISTERED per tale modulo. Prima di iniziare uno scambio dati con un WTIM, l'NCAP deve invocare un comando di tipo open, con il quale passa appunto allo stato OPEN per tale WTIM e vi rimane fino al primo comando di tipo close, attraverso il quale ritorna allo stato TIM REGISTERED. Il vantaggio di mantenere separati gli stati per ogni entità 1451.5 e per ogni WTIM consiste nel fatto che esso può iniziare in qualsiasi momento la ricerca di altri dispositivi e l'attesa di nuove connessioni.
Anche il funzionamento del WTIM è descritto per mezzo di una macchina a stati in cui a ogni stato corrispondono una o più azioni in base agli stimoli d'ingresso del WTIM stesso. Il funzionamento di tale macchina a stati è indicato in figura 1.5:

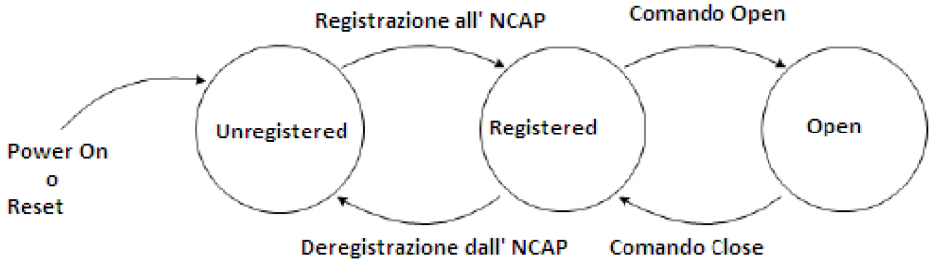

Figura 1.5: Macchina a stati WTIM

Lo stato iniziale di un WTIM dopo l'accensione o dopo il reset è UNREGISTERED. Dopo il processo di registrazione con un NCAP, il WTIM passa allo stato REGISTERED, ma non può comunicare finché non invoca un comando di tipo *open*, con cui passa appunto allo stato OPEN e vi rimane fino al primo comando di tipo *close*, attraverso il quale ritorna allo stato REGISTERED.

Capitolo 2

Standard ZigBee IEEE 802.15.4

Vediamo ora, in maniera più approfondita, il protocollo ZigBee. Questo protocollo si presta particolarmente a essere utilizzato in sensori a basso costo e a basso consumo di energia. In applicazioni di monitoraggio e controllo (spesso in campo industriale, ma non solo) queste due importanti caratteristiche permettono di creare vaste mesh network, con sensori *stand-alone* alimentati per molto tempo da batterie di dimensioni e costi contenuti. ZigBee è un protocollo ad alto livello basato, a livello fisico, sullo standard IEEE 802.15.4. Esso opera sulle frequenze assegnate per scopi industriali, medici e scientifici. Questa tecnologia risulta essere più semplice di altre quali il *Bluetooth* o il *WiFi* per reti Wireless Public Area Network (WPAN). Infatti un nodo di una rete ZigBee di elevata complessità può richiedere il 50% del codice rispetto a un nodo sviluppato con tecnologia *Bluetooth* o *WiFi*. Questo comporta un notevole risparmio di risorse in termini di memoria non volatile, necessaria per implementare lo stack protocollare.

2.1 Stack ZigBee

Uno stack è formato da un insieme di strati chiamati layer. Ogni layer fornisce un insieme di servizi al layer superiore, sfruttando a tale scopo alcuni servizi forniti dal layer sottostante. Ogni layer presenta delle interfacce attraverso le quali può comunicare con gli altri due layer adiacenti e una simile comunicazione si svolge attraverso una serie di primitive definite dagli stessi standard. L'architettura dello stack ZigBee è basata sullo standard OSI (Open System Interconnection), ma definisce solo i layer necessari per le funzionalità applicative richieste.

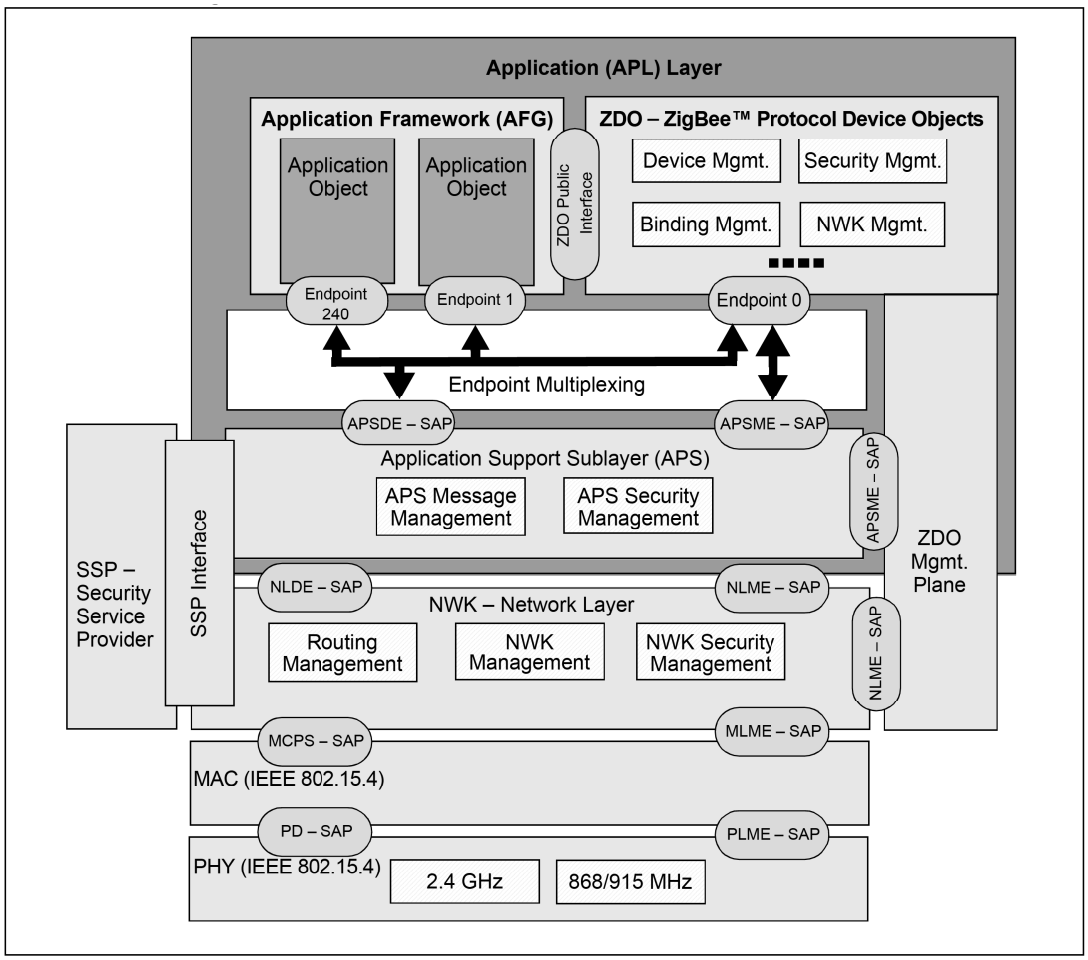

Figura 2.1: Architettura di uno Stack ZigBee

Lo standard IEEE 802.15.4 definisce i due layer più bassi: il PHY (physical layer) e il MAC (medium access control sub-layer). Sopra di essi sono stati costruiti il Network layer e il framework per l'Application layer che include tutto il necessario per lo sviluppo delle applicazioni su ZigBee. Lo standard a livello fisico si occupa della gestione delle risorse, come ad esempio la ripartizione della banda di frequenza disponibile. Le bande definite da IEEE 802.15.4 sono tre, collocate rispettivamente a 2.4 GHz, 915 MHz e 868 MHz e con un numero stabilito di canali. La banda a 2.4 GHz ha 16 canali, numerati da 11 a 26. Il bit rate dipende dalla frequenza operativa: a 2.4 GHz si raggiunge il bit rate massimo, pari a 250 kbps, mentre con il diminuire della frequenza diminuisce anche il bit rate. Il livello MAC svolge altri compiti, come per esempio il controllo degli errori, e in genere implementa le tecniche necessarie per garantire l'adattabilità della comunicazione. I pacchetti a livello MAC hanno al loro interno 16 bit di CRC che consentono di verificare l'integrità del pacchetto trasmesso. Quando il pacchetto è stato correttamente trasmesso viene inviata al mittente una conferma (Acknowledgement) dell'avvenuta ricezione. Se dopo un certo tempo non si riceve nessuna conferma, il mittente ritenta la trasmissione del

pacchetto. Bisogna prestare attenzione poiché questa verifica viene fatta a livello MAC. Se per qualche motivo il pacchetto non venisse accettato dai layer superiori, viene perso. I layer superiori richiedono quindi un ulteriore sistema di riconoscimento per l'avvenuta trasmissione dei pacchetti.

Il network layer si occupa delle procedure per la configurazione della rete, applica le misure di sicurezza ai frame e gestisce il routing. Per quanto concerne il routing ogni nodo della rete contiene una tabella (neighbor table) nella quale registra i nodi adiacenti della rete. Questa tabella viene aggiornata a livello di network layer quando nuovi nodi si connettono e si identificano. Se il nodo è un coordinatore, il network layer di tale nodo gestisce anche le procedure di creazione della rete e assegna gli indirizzi ai nodi connessi. Quando un nodo si connette il coordinatore gli assegna un identificativo univoco con cui verrà riconosciuto.
Lo standard ZigBee supporta vari tipi di rete: reti a stella, mesh e ad albero. La rete a stella è controllata da un singolo nodo chiamato coordinatore, necessario per creare e mantenere i nodi nella rete. Questi sono chiamati nodi finali (End Device) e comunicano direttamente con il coordinatore.
Di seguito è fornito un esempio di rete a stella:

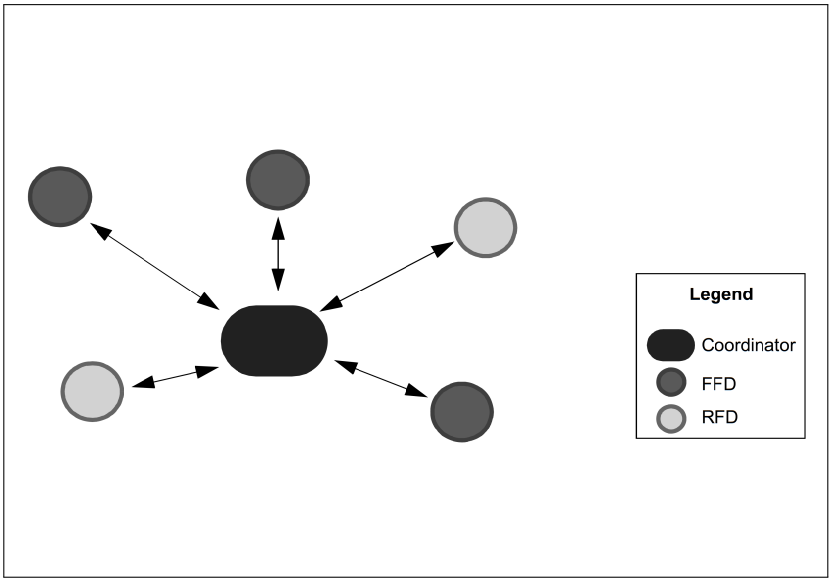

Figura 2.2: Esempio di una rete a stella

Nelle reti mesh o in quelle ad albero il coordinatore si occupa soltanto di inizializzare la rete. La rete può essere espansa con l'inserimento di nodi Router che gestiscono il traffico senza doverlo centralizzare nel coordinatore.

Di seguito sono riportati due esempi di rete ad albero e rete mesh:

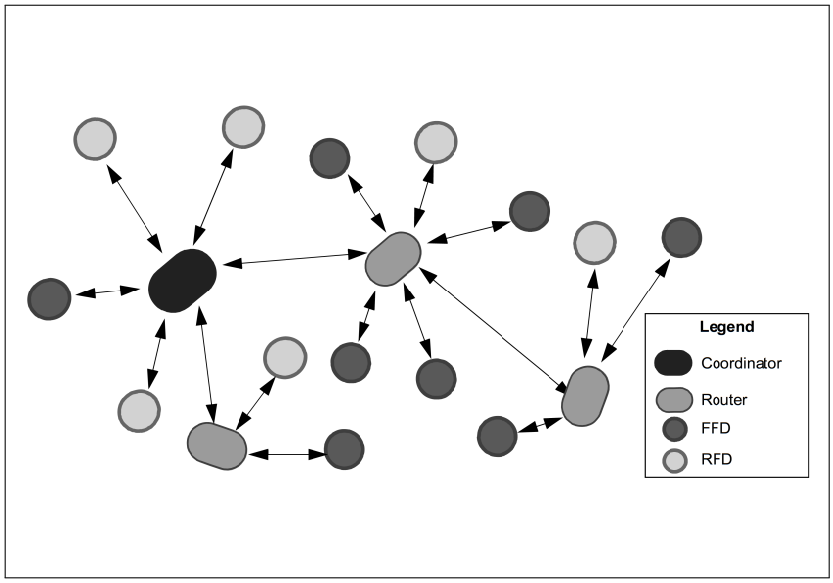

Figura 2.3: Esempio di una rete ad albero

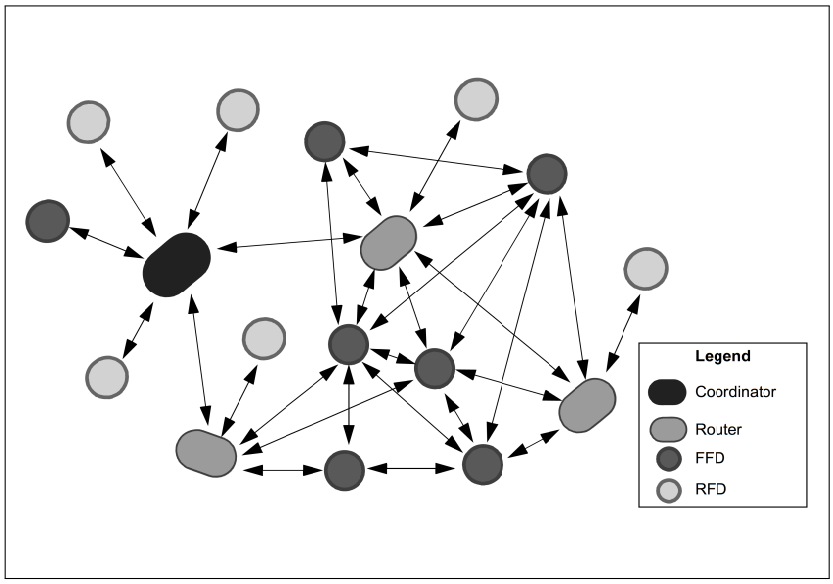

Figura 2.4: Esempio di una rete mesh

2.2 La rete: formazione e comunicazione tra nodi

In questa tesi verranno considerati soltanto due tipi di nodo ZigBee: il coordinatore e il nodo finale (End Device).
Il coordinatore è unico per ogni network e svolge i compiti di formare la rete, assegnare gli

indirizzi e mantenere la tabella di connessioni (binding table). Secondo quanto stabilito nel IEEE 802.15.4, i nodi finali possono essere di due tipi: FFD (Full Function Device) e RFD (Reduced Function Device). I nodi finali utilizzati in questo lavoro sono RFD, presentano quindi funzioni limitate e alimentazione a batteria e sono orientati a svolgere le loro funzioni in un regime di risparmio energetico.
Nella rete esistono due possibili meccanismi di accesso multiplo che regolano la comunicazione tra più utenti: beacon e non-beacon.
In una rete **beacon** i nodi sono autorizzati a trasmettere solo in time slot predefiniti. Il coordinatore periodicamente invia un superframe identificato come segnale di beacon con cui tutti i nodi devono sincronizzarsi. A ogni nodo è assegnato uno specifico slot durante il quale gli è consentito trasmettere e ricevere dati. La versione dello Stack Microchip supporta solo reti non-beacon.
In una rete **non-beacon** tutti i nodi sono autorizzati a trasmettere in ogni momento finché il canale è libero. Dopo aver rilevato che il canale è libero il dispositivo può trasmettere il pacchetto. Se il nodo destinatario è un FFD, allora il suo transceiver è sempre acceso e può comunicare in ogni momento. Questa caratteristica è presente, ad esempio, in una mesh network. Tuttavia se il nodo destinatario è un RFD, allora esso spegne il transceiver quando è inattivo per conservare energia. In questo stato il nodo non può ricevere messaggi e così è necessario che i messaggi in ingresso e in uscita dal nodo RFD passino per un nodo genitore FFD. Quando il nodo RFD torna attivo richiede i messaggi al suo nodo genitore: se questo ha nel suo buffer un messaggio per il proprio figlio, glielo inoltra. Ciò permette al nodo RFD di conservare energia, ma richiede la presenza nel nodo genitore di una memoria di buffer sufficiente a conservare i messaggi per ognuno dei figli. Se il nodo figlio non richiede i messaggi entro un certo tempo, allora il messaggio scade e il nodo genitore lo cancella.
Dopo aver brevemente descritto il modo in cui comunicano i nodi ZigBee, si analizza adesso il modo in cui viene formata una nuova rete.
In principio una nuova rete ZigBee è creata da un nodo coordinatore. All'accensione il coordinatore cerca altri coordinatori presenti nei canali in cui esso è autorizzato a trasmettere. In base alla potenza e alle reti presenti in ogni canale, esso crea una nuova rete con un identificativo univoco (16 bit PAN ID). Una volta creata la rete, i nodi finali sono abilitati a entrare nella rete. Ogni device di una rete ZigBee conserva nella sua neighbor table le informazioni riguardanti gli altri nodi della rete, i nodi genitori e figli, in una parte di memoria non volatile. Se all'accensione un nodo figlio tramite la sua neighbor table riconosce di esser stato parte di una rete, esso invia una notifica di nodo orfano agli altri nodi per ritornare a far parte della rete a cui era connesso in precedenza. I dispositivi che

ricevono la notifica del nodo orfano controllano nella loro tabella se il nodo risulta essere uno dei loro figli: se è così, il nodo genitore conferma al nodo figlio la sua precedente posizione nella rete. Se la notifica fallisce il nodo cerca di accedere come nuovo nodo nella rete. Una volta all'interno della rete un nodo può richiedere di uscire dalla stessa.

2.3 Terminologia e convenzioni

Lo standard ZigBee prevede una serie di convenzioni e terminologie necessarie a comprendere il funzionamento e le peculiarità della comunicazione. Il profilo di un protocollo ZigBee (profile) è una semplice descrizione dei componenti logici (device) e delle loro interfacce; non c'è codice associato a un profilo. Ogni dato che viene scambiato tra due device è un attributo (attribute), a ognuno dei quali è assegnato un identificativo univoco. Gli attributi sono raggruppati in clusters, a cui, a loro volta, viene assegnato un identificativo univoco. Il profilo stabilisce i valori degli identificativi (ID) per gli attributi e per i cluster, definendo quali sono obbligatori e quali opzionali; esso stabilisce inoltre se ci sono servizi opzionali del protocollo ZigBee da inserire come obbligatori in quella determinata applicazione. Ogni blocco funzionale definito all'interno del profilo è un endpoint che può contenere vari clusters. Ogni device comunica attraverso i suoi endpoint e i cluster abilitati.

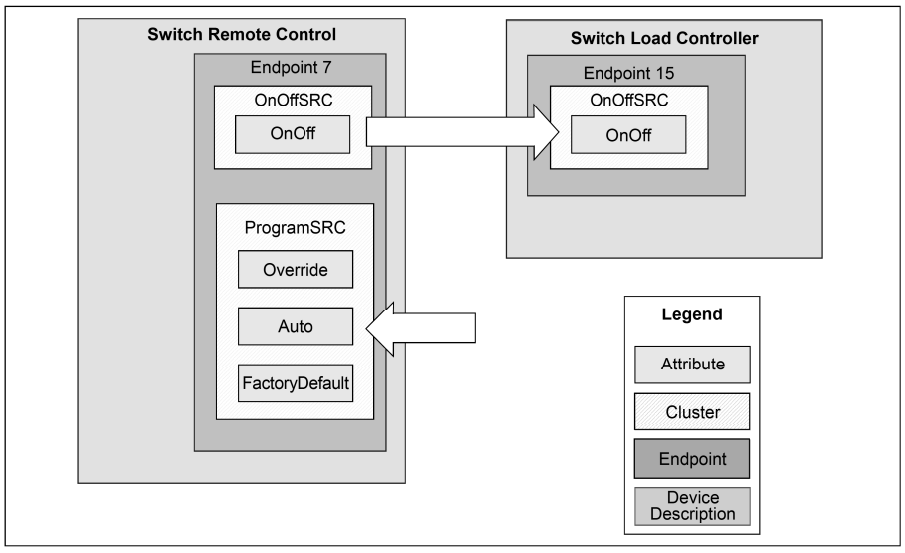

Figura 2.5: Architettura di un profilo del protocollo ZigBee

Ogni device può comunicare quindi con un altro nella stessa rete, se conosce il suo indirizzo; questo avviene quando si invia un messaggio diretto. Per semplificare la messaggistica

all'interno di una stessa rete il protocollo ZigBee utilizza il binding. Il coordinatore della rete crea una tabella delle corrispondenze a livello di cluster/endpoint tra i servizi e le richieste dei device. Ogni coppia di corrispondenza è chiamata binding. Una volta che il binding è stato creato, due devices possono comunicare attraverso il coordinatore. Il messaggio viene inviato al coordinatore, che lo inoltra a uno o più nodi presenti nella rete. Questi tipi di messaggi sono chiamati messaggi indiretti.

2.4 Caratteristiche tecniche

I dispositivi ZigBee devono rispettare le norme dello standard IEEE 802.15.4-2003 per Low-Rate Wireless Personal Area Network (WPAN). Esso specifica il protocollo di livello fisico (PHY) e il sottolivello Data Link del Medium Access Control (MAC).

PHY

Il protocollo ZigBee opera nella banda ISM non regolamentata, alle frequenze di 2.4 GHz, 915 MHz e 868 MHz. Nella banda 2.4 GHz ci sono 16 canali ZigBee da 3 MHz ciascuno. I trasmettitori radio usano una codifica DSSS. Si usa una modulazione BPSK nelle bande 868 e 915 MHz e una QPSK con offset (O-QPSK) che trasmette 2 bit per simbolo nella banda 2.4 GHz. Il data rate over-the-air è di 250 Kb/s per canale nella banda 2.4 GHz, 40 Kb/s per canale nella banda 915 MHz e 20 Kb/s nella banda 868 MHz. Il range di funzionamento è compreso tra 10 e 75 metri, a seconda dell'ambiente circostante. La massima potenza trasmessa è in genere 0 dBm (1 mW).

MAC

La modalità base di accesso al canale, specificata da IEEE 802.15.4-2003, è il Carrier Sense Multiple Access/Collision Avoidance (CSMA/CA). Questo significa che i nodi controllano se il canale è libero quando devono trasmettere. Vi sono alcune eccezioni all'uso del CSMA: i segnali di beacon - inviati secondo uno schema prefissato - i messaggi di acknowledge e le trasmissioni di dispositivi in reti beacon-oriented che necessitano una bassa latenza e usano Guaranteed Time Slots (GTS) che per definizione non fa uso di CSMA.
La lunghezza massima dei pacchetti MAC definiti dall'IEEE 802.15.4 è di 127 byte, compreso un campo CRC a 16 bit per il controllo dell'integrità del frame; inoltre lo standard IEEE 802.15.4 prevede l'uso (opzionale) di un meccanismo di acknowledge tramite l'im-

postazione di un flag di ACK all'interno dei frame inviati.
Un messaggio ZigBee può essere quindi formato al più da 127 byte, così suddivisi:

- Medium Access Control (MAC) header: contiene i campi di controllo del frame a livello MAC, il Beacon Sequence Number (BSN) e le informazioni sull'instradamento del messaggio (tale header è trasparente al livello applicazione, pertanto esso non verrà interessato da questo progetto);
- Network layer (NWK) header: contiene, tra le altre informazioni, l'indirizzo della sorgente e della destinazione (tale header è trasparente al livello applicazione, pertanto esso non sarà incluso in questo progetto);
- Application Support Sub-Layer (APS) header: contiene informazioni riguardanti il profilo, il cluster e l'endpoint di destinazione;
- APS payload: dati utili, la cui gestione spetta al livello applicazione.

Capitolo 3

Microchip ZigBee

3.1 Caratteristiche principali

Lo stack ZigBee fornito da Microchip che verrà utilizzato per lo sviluppo di questo progetto è basato sulla versione 2.6 del protocollo ZigBee e presenta le seguenti caratteristiche:

- supporto certificato della versione 2.6 del protocollo ZigBee;
- supporto della banda di frequenze a 2.4 GHz;
- supporto di tutti i tipi di dispositivi ZigBee (Coordinator, Router e End Device);
- struttura modulare e nomenclatura corrispondente a quella usata nel protocollo ZigBee e nelle specifiche IEEE 802.15.4;
- portabilità su tutte le famiglie di microcontrollori PIC18 e PIC24;
- supporto per l'indirizzamento multi casting;
- supporto per meccanismi di re-join degli End Device.

Esistono però alcune limitazioni, tra cui:

- supporto per sole reti non-beacon;
- indirizzo di rete non riassegnabile ai nodi che lasciano la rete;
- risoluzione dei conflitti PAN ID non supportata;
- frammentazione non supportata.

3.2 Architettura dello Stack

Il Microchip ZigBee Stack è stato progettato rispettando il protocollo ZigBee e le specifiche IEEE 802.15.4, mantenendo cioè la nomenclatura il più possibile coerente e organizzando i vari layer in file sorgenti separati, così da ottenere una libreria modulare indipendente dall'applicazione.
Il suddetto stack è scritto in linguaggio ANSI C ed è destinato ai microcontrollori Microchip della serie PIC18 e PIC24 montati a bordo del sistema di sviluppo PICDEM Z di Microchip, sfruttando dunque la memoria flash interna del processore per il salvataggio di parametri quali l'indirizzo MAC, la tabella dei vicini e la tabella di binding.
Il Microchip ZigBee Stack è progettato in modo da essere facilmente trasferibile in altri sistemi basati su microcontrollore PIC e in modo da supportare diversi transceiver con cambiamenti minimi ai livelli più alti dell'applicazione, senza dover intervenire nei layer più bassi. Infine il Microchip ZigBee Stack è ideato per funzionare con i compilatori Microchip MPLAB Compiler, ma con piccole modifiche può supportare anche altri compilatori di linguaggio ANSI C.
Lo stack si basa sul modello ISO/OSI, in cui ogni livello opera indipendentemente dagli altri fornendo ai livelli immediatamente adiacenti dei servizi su richiesta e scambiando dati attraverso i Service Access Point (SAP) predefiniti. In sostanza lo stack ZigBee messo a punto da Microchip è ridotto a quattro strati rispetto ai tradizionali sette previsti dal modello ISO/OSI. Con un approccio di tipo bottom-up si incontrano i seguenti layer:

- Physical Layer (PHY): definito nello standard IEEE 802.15.4, si occupa della gestione del transceiver wireless;
- Medium Access Control layer (MAC): definito nello standard IEEE 802.15.4, si occupa dell'accesso al mezzo;
- Network Layer (NWK): si occupa del routing (se previsto), della gestione e della sicurezza della rete;
- Application Layer (APL): unisce quelli che nel modello ISO/OSI sono i layer transport, session, presentation e application. Questo livello è perciò molto complesso e costituisce il cuore dello stack poiché, tramite il proprio sottolivello Application Support Sublayer (APS), scambia messaggi con i livelli sottostanti. All'interno del livello APL esiste inoltre un multiplexer che smista i messaggi inoltrandoli agli endpoint appropriati, siano essi gestiti dal framework dell'applicazione (AFG) oppure endpoint previsti dal protocollo ZigBee (ZDO).

Naturalmente a livello dell'applicazione deve essere presente anche il programma definito dall'utente.

File

L'installazione del Microchip ZigBee Stack copia nel sistema, oltre ai file sorgenti di alcuni esempi funzionanti, i file sorgenti dello stack protocollare, che sono collocati nelle directories ZigBeeStack e Common.

Funzionamento Generico

Innanzitutto il codice sorgente dell'applicazione deve includere il file header zAPL.h per poter avere accesso alle funzioni dello stack ZigBee. Ogni dispositivo poi deve necessariamente memorizzare una variabile di tipo ZIGBEE PRIMITIVE (qui identificata con il nome currentPrimitive) per conservare traccia in ogni momento della primitiva in esecuzione da parte dello stack. I dispositivi Router e End Device inoltre devono memorizzare le variabili currentNetworkDescriptor e NetworkDescriptor del tipo NETWORK DESCRIPTOR utilizzate per le operazioni di rete; una volta che queste variabili sono state memorizzate, l'applicazione deve configurare i registri e i pin del microprocessore per attivare l'interfaccia con il transceiver. Ora lo stack può essere inizializzato con la chiamata alla funzione ZigBeeInit() e possono essere attivati gli interrupt con le istruzioni RCONbits.IPEN = 1 e INTCONbits.GIEH = 1.
Completate le procedure appena descritte, lo stack può funzionare attraverso la gestione delle primitive definite dai protocolli ZigBee e IEEE 802.15.4. Dopo la memorizzazione del nome della primitiva da eseguire nella variabile currentPrimitive, la chiamata alla funzione ZigBeeTasks() innesca la procedura di funzionamento dello stack per la primitiva in questione. Tale funzione infatti è un handler (gestore) delle primitive, ovvero si occupa di coordinare i vari layer passando loro le primitive da gestire, anche in maniera ricorsiva. Il meccanismo automatico cessa quando una primitiva ha esaurito il suo percorso di gestione oppure quando deve essere processata dal livello applicazione. Ogni volta che termina il ciclo di gestione di una primitiva la variabile currentPrimitive deve essere aggiornata con la successiva primitiva da gestire oppure il sistema può essere lasciato in attesa tramite l'assegnazione della primitiva speciale NO PRIMITIVE. Poiché può essere gestita soltanto una primitiva per volta, esiste una struttura dati (descritta nelle ZigBeeTasks.h) adatta al mantenimento dei parametri relativi alla primitiva in corso di gestione. Lo stack dà inoltre la possibilità di visualizzare per via seriale l'output dell'applicazione attraverso un

terminale; per fare ciò la porta seriale deve essere configurata con i parametri: 19200 b/s, 8 bit di dati, 1 bit di stop, senza controllo di parità né di flusso (19200,N,8,1).

Primitive ZigBee principali

La tabella che segue contiene l'elenco delle primitive e delle relative risposte utilizzate per le principali funzionalità dello stack ZigBee (ovviamente non tutti i dispositivi implementeranno queste primitive).

Funzioni principali

Lo stack ZigBee mette a disposizione dell'utente alcune funzioni utili alla semplificazione dell'applicazione. Di seguito l'elenco dettagliato delle funzioni utilizzate:

APLDisable: questa funzione disabilita il ricetrasmettitore ed è solitamente usata dai dispositivi di tipo RFD per risparmiare energia nella modalità sleep mode:

- sintassi: BOOL APLDisable(void);
- input: nessuno;
- output: TRUE se il ricetrasmettitore è stato disabilitato, FALSE se è stato impossibile disabilitarlo.

APLDiscard: questa funzione, che deve essere chiamata al termine del processamento di ogni messaggio ricevuto, scarta il messaggio corrente. Il fallimento di tale funzione impedisce il processamento e la ricezione di ulteriori messaggi:

- sintassi: void APLDiscard(void);
- input: nessuno;
- output: nessuno.

APLEnable: questa funzione abilita il ricetrasmettitore:

- sintassi: void APLEnable(void);
- input: nessuno;
- output: nessuno.

APLGet: questa funzione viene utilizzata dall'applicazione per ricevere un byte del messaggio in corso di processamento da parte dei livelli sottostanti. Se chiamata dopo che tutto il messaggio è stato ricevuto, restituisce il valore 0x00. Il puntatore al byte corrente si aggiorna in modo automatico ad ogni chiamata:

- sintassi: BYTE APLGet(void);
- input: nessuno;
- output: il byte corrente del messaggio in corso di processamento.

ZigBeeBlockTx: questa funzione blocca il buffer di trasmissione (TxBuffer). Per avere la conferma che il buffer sia effettivamente bloccato, la successiva chiamata alla funzione ZigBeeReady deve restituire il valore FALSE:

- sintassi: void ZigBeeBlockTx(void);
- input: nessuno;
- output: nessuno.

ZigBeeInit: questa funzione inizializza lo stack ZigBee e dev'essere invocata prima di qualsiasi altra funzione e dopo la configurazione dell'hardware:

- sintassi: void ZigBeeInit(void);
- input: nessuno;
- output: nessuno.

ZigBeeReady: questa funzione indica se lo stack è pronto all'invio di un messaggio:

- sintassi: BOOL ZigBeeReady(void);
- input: nessuno;
- output: TRUE se è possibile caricare un nuovo messaggio nel buffer d'uscita, FALSE se il buffer è ancora occupato dal messaggio precedente.

ZigBeeTasks: questa funzione coordina le operazioni dello stack. Il riferimento alla primitiva da eseguire deve essere passato nella variabile **primitive* (se non ci sono primitive da eseguire occorre riferirsi alla primitiva NO PRIMITIVE).
La funzione continuerà finché non ci saranno primitive del livello applicazione da eseguire:

- sintassi: BOOL ZigBeeTasks(ZIGBEE PRIMITIVE *primitive);
- input: primitive, puntatore al valore della prossima primitiva da eseguire;
- output: TRUE se lo stack ha ancora dei task da eseguire in background, FALSE se non ne ha.

Status Flag

Lo stack ZigBee possiede diversi status flag a disposizione dell'applicazione, che tuttavia non devono essere modificati, al fine di scongiurare malfunzionamenti alle operazioni dello stack. Tutti i flag sono contenuti nella struttura ZigBeeStatus.ags.bits.

3.3 Creazione e connessione a una rete

Per creare una rete il nodo coordinatore deve eseguire la primitiva NLME - NETWORK FORMATION request. Se il dispositivo non è un coordinatore e non è connesso a nessuna rete, esso deve raggiungerne una; nel caso fosse stato precedentemente connesso a una rete deve tentare di riconnettersi alla stessa, utilizzando la primitiva NLME JOIN request con il parametro RejoinNetwork settato a TRUE. Se questa procedura fallisce oppure nel caso il dispositivo non facesse precedentemente parte di una rete, esso deve tentare di connettersi come nuovo nodo e per farlo deve innanzitutto scoprire le reti disponibili attivando la primitiva NLME NETWORK DISCOVERY request. Dopo ciò l'applicazione sceglie la rete a cui collegarsi tramite la primitiva NLME JOIN request, questa volta con il parametro RejoinNetwork settato a FALSE.

Invio di un messaggio

Lo stack ZigBee implementato da Microchip permette di inviare un solo messaggio per volta. Perché ciò sia possibile è innanzitutto necessario verificare che la funzione ZigBeeReady() restituisca il valore TRUE, a indicare che lo stack è pronto. La funzione ZigBeeBlockTx() blocca le trasmissioni (di conseguenza ulteriori chiamate a ZigBeeReady() restituiscono il valore FALSE) ed è quindi possibile caricare il payload del messaggio da inviare nel vettore TxBuffer (indicizzato dalla variabile TxData), nonché i parametri d'invio nelle apposite locazioni di memoria. Una volta terminato il caricamento la variabile TxData deve puntare al primo elemento libero del buffer, cosicché TxData indichi anche la lunghezza dei dati in esso contenuti. Successivamente è necessario settare currentPrimitive con la primitiva APSDE DATA request, caricarne i relativi parametri e chiamare

la funzione ZigBeeTasks(). Per inviare un messaggio si devono conoscere l'indirizzo e l'endpoint della destinazione, che verranno salvati rispettivamente nelle variabili destinationAddress e destinationEndpoint. Poiché ogni frame è identificato da un Transaction ID univoco, esso può essere reperito richiamando la funzione APLGetTransID(). Lo stato di invio del messaggio è restituito dalla primitiva APSDE DATA confirm; nel caso l'invio fallisse, lo stack lo ritenta automaticamente, fino a un massimo di tentativi specificato dalla variabile apscMaxFrameRetries.

Ricezione di un messaggio

Lo stack notifica all'applicazione la ricezione di un nuovo messaggio attraverso la primitiva APSDE DATA indication. Assieme a questa primitiva lo stack fornisce tutte le informazioni e i parametri relativi al messaggio memorizzato nel buffer. A ogni chiamata la funzione APLGet() estrae sequenzialmente un byte dal buffer. Il parametro DstEndpoint, come rivela il nome stesso, indica l'endpoint di destinazione. Se il messaggio pervenuto all'endpoint è corretto può essere processato, altrimenti deve essere scartato.
Dopo essere stato processato il messaggio deve essere comunque scartato tramite la funzione APLDiscard(), in modo da consentire anche ai messaggi successivi di essere processati. La mancata osservanza di questa regola comporta il blocco della ricezione dei messaggi dopo il primo messaggio processato ma non scartato.

Sincronizzazione dell'RFD

L'RFD solitamente funziona in modalità risparmio energetico e per far ciò spegne il transceiver quando è inattivo (sleep mode). Lo sleep mode si attiva solo se sono verificate tutte le seguenti condizioni:

- non ci devono essere primitive da processare;
- lo stack non deve avere tasks da eseguire in background;
- la precedente richiesta di dati deve essere stata completata;
- tutti i processi specifici gestiti dall'applicazione devono essere terminati.

Lo sleep mode può essere interrotto dal verificarsi di una condizione di interrupt oppure dalla scadenza del watchdog timer. Al risveglio, dopo aver acceso il ricetrasmettitore, l'RFD deve richiedere al proprio nodo genitore l'inoltro dei messaggi arrivati nel lasso di tempo in cui era inattivo e a esso riservati. Questo è possibile tramite l'uso della primitiva NLME SYNC request, seguito dalla consueta chiamata all'handler ZigBeeTasks().

Anche nel caso in cui il nodo genitore abbia più messaggi da inoltrare, esso ne invia uno (il primo in ordine cronologico) e l'RFD di conseguenza attiva la procedura di ricezione tramite la primitiva APSDE DATA indication; nel caso in cui il nodo genitore non avesse messaggi da inoltrare, l'RFD genererebbe la primitiva NLME SYNC confirm accompagnata dallo status SUCCESS. Infine, nel caso peggiore in cui l'RFD non riceva alcuna risposta dal proprio nodo genitore, genera comunque la primitiva NLME SYNC confirm ma accompagnata dallo status NWK SYNC failure[1].

Recupero delle risorse inutilizzate

Dopo l'utilizzo di una primitiva è necessario che tutte le risorse obsolete siano cancellate. Il Microchip ZigBee Stack si occupa autonomamente di questo servizio, tranne nel caso in cui la primitiva appena processata sia NLME JOIN confirm. Tutti i nodi della rete, escluso il coordinatore, invocano la primitiva NLME - NETWORK DISCOVERY request per ricercare le reti disponibili. La primitiva NLME NETWORK DISCOVERY confirm restituisce quindi un puntatore a una lista di reti disponibili, che costituirà la base della decisione del nodo. Una volta selezionata e raggiunta la rete il nodo genera la primitiva NLME JOIN confirm, a indicare l'avvenuta connessione. A questo punto la lista delle reti disponibili diventa inutile e può essere cancellata dall'applicazione per liberare la memoria.

Temporizzazione ZigBee

Il data rate nella banda 2.4 GHz è di 250 Kbps. In ogni periodo di simbolo si inviano quattro bit, pertanto il periodo di simbolo è di 16 μs; la temporizzazione interna dello stack è basata sul periodo di simbolo. Sia le reti beacon che quelle non-beacon basano la loro temporizzazione su degli appositi frame chiamati superframe, sebbene questi non vengano utilizzati nelle reti non-beacon. La durata del superframe, definita dal parametro aBaseSuper-frameDuration, rappresenta il numero di simboli che compongono lo slot di un superframe (aBaseSlotDuration, 60) moltiplicato per il numero di slot contenuti in un superframe (aNumSuperframeSlots, 16). La durata della scansione è dovuta alle primitive NLME NETWORK DISCOVERY request, NLME NETWORK FORMATION request e NLME JOIN - request è di aBaseSuper-frameDuration $(2n + 1)$ simboli, dove n è il valore del parametro ScanDuration. Per il Microchip ZigBee Stack tale parametro può essere

[1]Nota: questo progetto prevede che l'NCAP e il WTIM rimangano sempre attivi mantenendo costantemente acceso il rispettivo ricetrasmettitore (rete non-beacon enabled), risulta dunque ininfluente valutare la sincronizzazione dell'RFD, che quindi non verrà considerata.

compreso tra 0 e 14, limitando l'intervallo di scan in un range compreso tra 0.031 secondi e 4.2 minuti[1].

3.4 Module Communications API - P2PComm

È l'interfaccia che collega il modulo 1451.5 allo strato ZigBee e rappresenta il nucleo dello strato di convergenza tra i due protocolli in quanto, tramite le funzioni qui implementate, si va a trasferire il payload IEEE1451 al sistema fisico di trasmissione che è ZigBee.

write: la funzione write trasmette il payload dal modulo IEEE 1451.5 a ZigBee che lo invia al TIM specificato nell'indirizzo (impostato dalla funzione *open*()). Questo avviene per mezzo di un ciclo *for* che carica il buffer di trasmissione TxBuffer byte a byte con il Command Message. Successivamente viene impostata la variabile currentPrimitive con la primitiva APSDE_DATA_request, la cui funzione è preparare e avviare una comunicazione radio. Alla successiva iterazione del ciclo macchina, ZigBee processerà tale primitiva e trasmetterà il contenuto del buffer di trasmissione al WTIM, che risponderà con un Reply Message contenente i dati di potenza ed energia letti dal sensore, e la funzione read() trasmetterà successivamente il dato al modulo 1451.5. La caratteristica di tale funzione è che l'elaborazione della primitiva APSDE_DATA_request non avviene nel ciclo macchina principale, bensì all'interno della funzione stessa; ciò è possibile grazie all'utilizzo ricorsivo del ciclo macchina all'interno di read(). Questa soluzione ha il vantaggio di interfacciare gli standard ZigBee e IEEE 1451 senza dover modificare il programma principale dal momento che tutte le operazioni di lettura e scrittura avvengono su un'unità di traduzione diversa.

read: compie il procedimento inverso rispetto a write(); questo metodo viene invocato in ricezione per trasferire i dati di potenza ed energia da ZigBee al modulo IEEE 1451.5. Il dato viene prelevato dal buffer di ricezione, convertito in un formato alfanumerico e incapsulato nel payload IEEE 1451.

[1]Nota: questo progetto prevede che l'NCAP rimanga sempre attivo per richiedere le letture al WTIM; quest'ultimo manterrà sempre acceso il suo ricetrasmettitore (rete non-beacon enabled), dunque qui non si utilizzeranno i super-frame.

Capitolo 4

Sensore di potenza ed energia

4.1 Potenza ed energia in regime sinusoidale

Prima di affrontare lo studio del funzionamento della scheda sensore è opportuno richiamare la teoria alla base delle misurazioni che questa effettua.
Si consideri l'ipotesi di studiare una rete in regime sinusoidale: ponendo che la parte attiva che fornisce l'energia a un carico passivo sia modellizzabile come un generatore di tensione con funzione $v(t)$ si ha:

$$v(t) = V_M \sin(\omega t + \alpha) \quad [V] \tag{4.1}$$

dove:

- V_M è l'ampiezza massima (o di picco) della tensione [V];
- $\omega = 2\pi f$ è la pulsazione angolare [rad/s] e f è la frequenza [Hz];
- t è il tempo [s];
- α è la fase iniziale [rad].

Il **valore efficace** V_{RMS} per una tensione sinusoidale periodica è definito come:

$$V_{RMS} = \sqrt{\frac{1}{T}\int_T v^2(t)dt} = \frac{V_M}{\sqrt{2}} \quad [V] \tag{4.2}$$

e rappresenta l'equivalente valore di tensione continua tale per cui si produce lo stesso effetto termico su un carico resistivo.

In seguito si scriverà V inteso come V_{RMS}.
Considerando un generico bipolo passivo in regime sinusoidale possiamo scrivere:

$$\begin{cases} v(t) &= V_M \sin(\omega t + \alpha) \qquad [V] \\ i(t) &= I_M \sin(\omega t + \beta) \qquad [A] \end{cases} \tag{4.3}$$

il **ritardo di fase** tra tensione e corrente è:

$$\varphi = \alpha - \beta \tag{4.4}$$

la **potenza istantanea** si definisce come:

$$p(t) = v(t)i(t) = V_M \sin(\omega t + \alpha) I_M \sin(\omega t + \beta) \quad [VA] \tag{4.5}$$

La sua unità di misura è il voltampere [VA]; ricordando la formula trigonometrica di Werner[1] riscriviamo (4.5) e ricaviamo:

$$\begin{aligned} p(t) &= \frac{V_M I_M}{2}[\cos(\alpha - \beta) - \cos(2\omega t + \alpha + \beta)] = \\ &= VI\cos(\alpha - \beta) - VI\cos(2\omega t + \alpha + \beta) \quad [VA] \end{aligned} \tag{4.6}$$

Dall'ultima equazione si vede che la potenza istantanea è formata da due componenti: una continua nel tempo P_{cost} e l'altra, chiamata **potenza fluttuante** $p_f(t)$, che ha una pulsazione angolare pari a 2ω.
Riscriviamo quindi la potenza istantanea come somma di due componenti:

$$p(t) \quad = \quad P_{cost} \quad + \quad p_f(t) \tag{4.7}$$

con:

$$P_{cost} = VI\cos\varphi \tag{4.8}$$

$$p_f(t) = -VI\cos(2\omega t + \alpha + \beta) \tag{4.9}$$

Le grandezze di misura per la potenza sono:

- **potenza attiva (reale)**

$$P = \frac{1}{T}\int_T p(t)dt = VI\cos\varphi \quad [W] \tag{4.10}$$

  che corrisponde anche al valore medio ed è misurata in Watt;

[1] $\sin(x)\sin(y) = \frac{1}{2}[\cos(x-y) - \cos(x+y)]$

- **potenza reattiva (immaginaria)**

$$Q = VI \sin\varphi \quad [VAR] \tag{4.11}$$

che si misura in voltampere reattivi;

- **potenza apparente**

$$S = VI \quad [VA] \tag{4.12}$$

misurata in voltampere.

L'**energia** è definita come:

$$W = \int p(t)dt \quad [J] \tag{4.13}$$

ed è misurata in Joule.
Per completezza si definisce il **fattore di potenza**:

$$PF = \frac{P}{S} = \cos\varphi \tag{4.14}$$

Si ricordi che i valori minimi che questo numero può assumere per le utenze elettriche sono fissati per legge al fine di ridurre la dissipazione nelle linee elettriche dovuta alla parte reattiva della potenza. Per esempio un tipico valore minimo per un'utenza privata è PF = 0.9.
Il sensore di energia utilizzato rende disponibile la misura di potenza ed energia della sola parte attiva, trascurando completamente la componente reattiva.

4.2 La conversione A\D ΣΔ (Sigma-Delta)

La conversione ΣΔ (Sigma-Delta) è un tipo di conversione analogico - digitale che garantisce un alto livello di quantizzazione (fino a 20 bit) e che include tra i suoi vantaggi un'ottima linearità nella conversione e un alto rapporto S/N (segnale/rumore).
Il convertitore Sigma-Delta quantizza il segnale analogico in ingresso con una bassissima risoluzione e con una frequenza di campionamento molto elevata.
Le due parti fondamentali sono un modulatore Sigma-Delta che fornisce una sequenza seriale di bit con frequenza elevatissima e un filtro digitale che presenta in uscita dati digitali di frequenza inferiore, ma con risoluzione molto migliore.

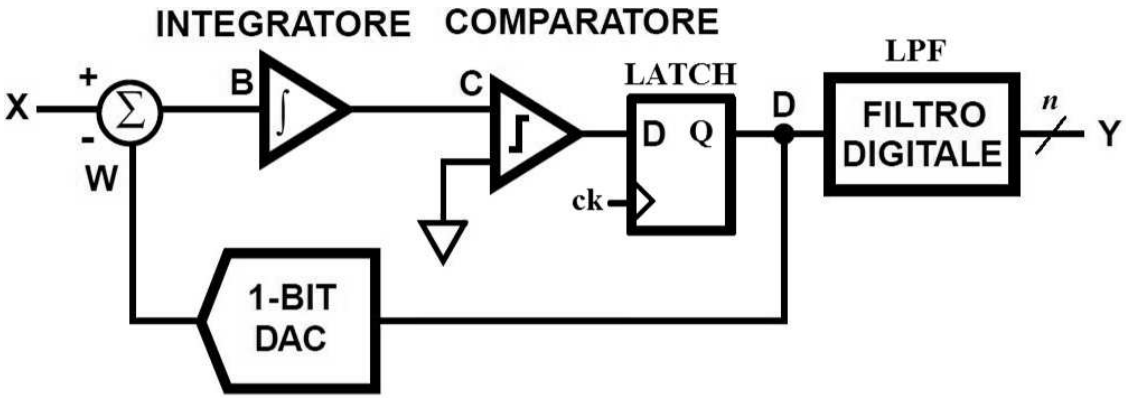

Figura 4.1: Schema di principio della conversione $\Sigma\Delta$

Dalla figura 4.1 si vede che l'ADC è composto da un nodo sommatore, da un integratore e da un convertitore A/D a un bit (cioè un comparatore), il tutto retroazionato negativamente da un convertitore D/A a un bit.
La presenza di un anello di retroazione impone che il sistema si porti in equilibrio solo quando la tensione $B = X - W$ è nulla in media, cioè quando la sequenza di bit prodotta dall'ADC ha media pari alla tensione presente in ingresso X.
Le operazioni sono sincronizzate da un segnale di clock (ck) e si ripetono con frequenza elevatissima f_C. All'uscita dell'ADC a un bit, e quindi del modulatore, si avrà una sequenza seriale di bit con cadenza f_C.
Il contenuto informativo del segnale d'ingresso V_{in} con larghezza di banda f_M è rappresentato dal duty cycle del treno di impulsi che si presenta all'uscita del modulatore.
La sequenza di bit viene trattata da un filtro passa basso digitale (LPF), che calcola il valore medio su un certo numero di campioni e fornisce dati digitali espressi in parole a n bit con frequenza $f_O < f_C$ a un registro di uscita.
Affinché sia rispettato il *teorema del campionamento* la frequenza f_O (output data rate) deve rispettare i seguenti vincoli:

$$f_C > 2f_O \tag{4.15}$$

$$f_O > 2f_M \tag{4.16}$$

Il rapporto $OSR = f_C/2f_M$ rappresenta il rapporto di sovracampionamento (oversampling): nella scheda Energy Meter vale 64 e la conversione viene effettuata a una frequenza di MCLK/4, cioè con un quarzo da 3.579 MHz vale $f_C \approx 900$ KHz, assicurando una banda passante di 5 KHz. Infatti, le normative richiedono di rilevare almeno le frequenze fino alla quinta armonica (250 Hz).

4.3 MCP3905A Evaluation Board

La scheda di valutazione a disposizione serve a misurare la potenza attiva e l'energia assorbita da un carico monofase.
Le informazioni fornite da questa scheda verranno gestite da un microcontrollore PIC18, realizzando così un WTIM che potrà inviare queste informazioni a un NCAP attraverso lo standard wireless ZigBee.
É opportuno descrivere in modo dettagliato il cuore della scheda, che è il circuito integrato MCP3905A della Microchip.

4.3.1 L'integrato Microchip MCP3905A

Il circuito integrato MCP3905A della Microchip è un misuratore di potenza attiva specifico per impianti monofase.

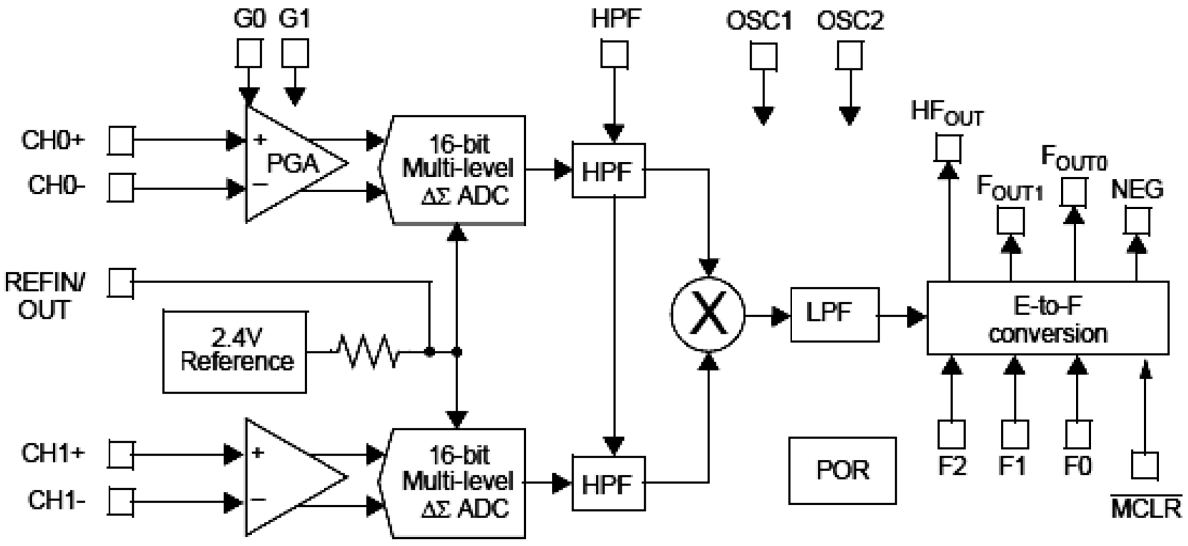

Figura 4.2: Schema a blocchi dell'integrato MCP3905A

Dallo schema a blocchi dell'integrato MCP3905A riportato in figura 4.2 si può notare la presenza di due ingressi differenziali adatti a misurare un segnale non riferito a massa, ai quali vanno collegati due segnali in tensione proporzionali alla tensione e alla corrente ai capi del carico.
L'integrato in questione presenta in uscita due tipi di segnale: il primo è un impulso periodico con frequenza dell'ordine delle decine di Hz (identificato in seguito con HF_{OUT}) che rappresenta la potenza attiva istantanea, mentre il secondo è un segnale con frequenza dell'ordine di frazioni di Hz (identificato in seguito con $F_{OUT0/1}$) pensato per comandare

direttamente un contatore di energia elettrica.
Il grafico in figura 4.3 illustra le forme d'onda in uscita:

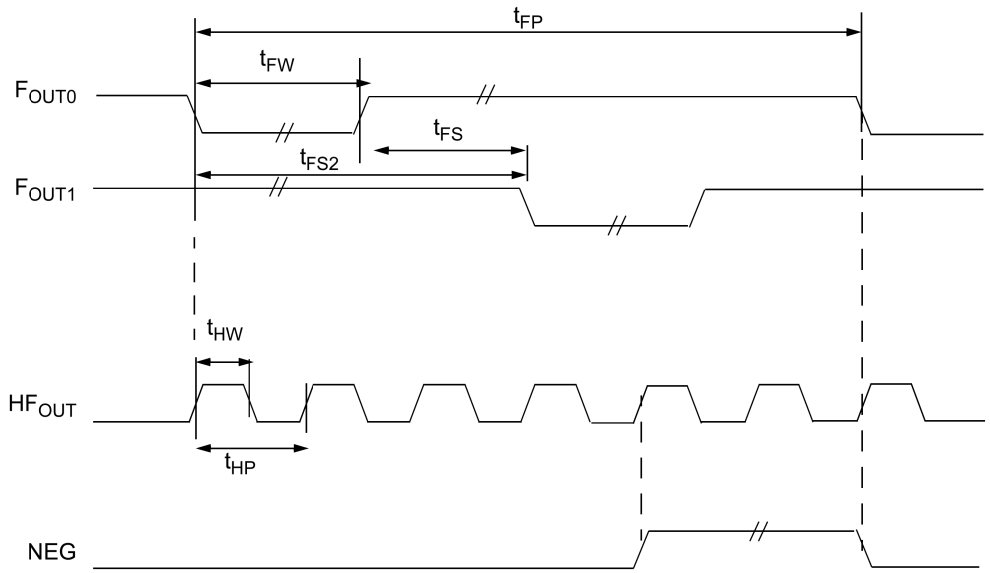

Figura 4.3: Forme d'onda in uscita dell'integrato MCP3905A

TIMING CHARACTERISTICS

Electrical Specifications: Unless otherwise indicated, all parameters apply at AV_{DD} = DV_{DD} = 4.5V – 5.5V, A_{GND}, D_{GND} = 0V, MCLK = 3.58 MHz; T_A = -40°C to +85°C.

Parameter	Sym	Min	Typ	Max	Units	Comment
Frequency Output						
F_{OUT0} and F_{OUT1} Pulse Width (Logic-Low)	t_{FW}	—	275	—	ms	984376 MCLK periods **(Note 1)**
HF_{OUT} Pulse Width	t_{HW}	—	90	—	ms	322160 MCLK periods **(Note 2)**
F_{OUT0} and F_{OUT1} Pulse Period	t_{FP}	Refer to Equation 4-1			s	
HF_{OUT} Pulse Period	t_{HP}	Refer to Equation 4-2			s	
F_{OUT0} to F_{OUT1} Falling-Edge Time	t_{FS2}	—	0.5 t_{FP}	—		
F_{OUT0} to F_{OUT1} Min Separation	t_{FS}	—	4/MCLK	—		
F_{OUT0} and F_{OUT1} Output High Voltage	V_{OH}	4.5	—	—	V	I_{OH} = 10 mA, DV_{DD} = 5.0V
F_{OUT0} and F_{OUT1} Output Low Voltage	V_{OL}	—	—	0.5	V	I_{OL} = 10 mA, DV_{DD} = 5.0V
HF_{OUT} Output High Voltage	V_{OH}	4.0	—	—	V	I_{OH} = 5 mA, DV_{DD} = 5.0V
HF_{OUT} Output Low Voltage	V_{OL}	—	—	0.5	V	I_{OL} = 5 mA, DV_{DD} = 5.0V
High-Level Input Voltage (All Digital Input Pins)	V_{IH}	2.4	—	—	V	DV_{DD} = 5.0V
Low-Level Input Voltage (All Digital Input Pins)	V_{IL}	—	—	0.85	V	DV_{DD} = 5.0V
Input Leakage Current		—	—	±3	µA	V_{IN} = 0, V_{IN} = DV_{DD}
Pin Capacitance		—	—	10	pF	**Note 3**

Note 1: If output pulse period (t_{FP}) falls below 984376*2 MCLK periods, then t_{FW} = 1/2 t_{FP}.
2: If output pulse period (t_{HP}) falls below 322160*2 MCLK periods, then t_{HW} = 1/2 t_{HP}.
3: Specified by characterization, not production tested.

Figura 4.4: Tabella di riferimento per le forme d'onda in uscita

L'integrato rende disponibile sul pin NEG un'altra informazione: questo pin sincronizzato con HF_{OUT} è a 1 logico nel momento in cui la differenza di fase è maggiore di 90°.
Questa informazione è utile perché amplia di molto l'applicazione di questo sistema; mi riferisco alle *smart grid*, cioè alle reti in cui non c'è più un solo generatore con molti utilizzatori passivi, ma gli stessi utilizzatori diventano attivi per esempio a opera di un impianto fotovoltaico. A questo punto sarebbe possibile progettare una rete di sensori in modo da monitorare la distribuzione della potenza nelle linee.

L'integrato dispone di una funzione aggiuntiva automatica che blocca gli impulsi in uscita quando il carico viene sconnesso; questo avviene al di sotto di un valore di corrente di soglia o start-up definito come lo 0.0015% del valore di frequenza d'uscita di fondo scala.

Analisi in frequenza

Si veda ora più in dettaglio ciò che avviene all'interno dell'integrato, facendo un'analisi in frequenza con riferimento alla figura 4.5:

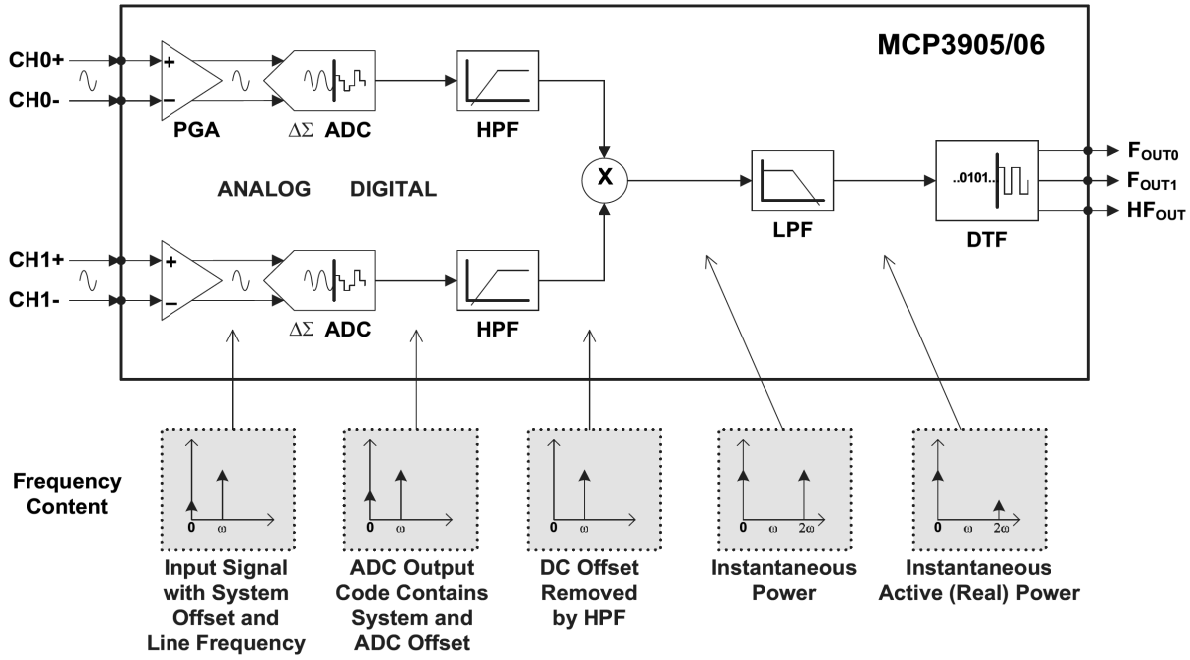

Figura 4.5: Analisi in frequenza dell'integrato MCP3905A

Si effettua una conversione analogico - digitale a 16 bit grazie a due convertitori A\D $\Sigma\Delta$ sui due segnali differenziali all'ingresso:

$$v(t) = V_M e^{j2\pi ft+\varphi} \quad \rightarrow \quad v(nT) = V_M e^{j2\pi fnT+\varphi} \tag{4.17}$$

I due convertitori A\D sono sincronizzati con il medesimo segnale di clock ed effettuano una conversione ogni MCLK/4; nel nostro caso la scheda ha a bordo un quarzo da 3.579 MHz e quindi la frequenza di campionamento vale $f_S \approx 900$ KHz.
È importante eliminare le frequenze superiori a quella di Nyquist cioè superiori a $f_S/2 \approx$ 450 KHz con un filtro anti-aliasing posto prima del convertitore A\D; nella scheda questo è realizzato con un filtro RC del secondo ordine.
L'integrato ha al suo interno un DSP (Digital Signal Processor) che moltiplica i due segnali digitali convertiti, ottenendo così la potenza istantanea.
Si noti che tra i convertitori A\D e il moltiplicatore vi sono due filtri digitali passa alto (HPF) che servono a eliminare un'eventuale componente continua di disturbo. All'uscita del moltiplicatore si ha:

$$p(nT) = v(nT)i(nT) = VI[\cos(\alpha - \beta) - \cos(2\omega nT + \alpha + \beta)] \tag{4.18}$$

A questo punto si deve mantenere la sola componente attiva della potenza e il modo più semplice per ottenere tale informazione consiste nel filtrare il segnale $p(nT)$ con un filtro passa basso (LPF), in modo da rimuovere la componente con pulsazione 2ω e lasciare la sola componente continua:

$$P = VI\cos(\alpha - \beta) \tag{4.19}$$

La componente con pulsazione 2ω è stata ridotta notevolmente dal filtro passa basso del primo ordine con frequenza di taglio 8.9 Hz; tuttavia, come riportano le note descrittive, nell'uscita HF_{OUT} essa non è trascurabile.
Il DSP converte il segnale all'uscita del filtro passa basso estraendo due informazioni differenti che rappresentano la **potenza attiva istantanea** e l'**energia**; queste informazioni sono contenute nella frequenza (DTF) dei due segnali rettangolari all'uscita dell'integrato.
L'integrato in questione deve essere programmato in base alle esigenze, generalmente rappresentate dal consumo massimo di energia del carico in questione.
A tal fine sono presenti degli ingressi di configurazione F2, F1, F0, G1 e G0 per scalare opportunamente le frequenze dei segnali in uscita, in base al consumo massimo di potenza del carico collegato.
Si riportano qui di seguito le formule dei segnali in uscita, in funzione dei parametri il cui valore è definito dai pin di configurazione:

$$F_{OUT0/1} = \frac{8.06 \quad V_0 \quad V_1 \quad G \quad F_C}{(V_{REF})^2} \quad [Hz] \tag{4.20}$$

$$HF_{OUT} = \frac{8.06 \quad V_0 \quad V_1 \quad G \quad HF_C}{(V_{REF})^2} \quad [Hz] \tag{4.21}$$

dove

- V_0 è la tensione V_{RMS} differenziale al canale CH0 (corrente);
- V_1 è la tensione V_{RMS} differenziale al canale CH1 (tensione);
- G è il guadagno sul canale CH0 (corrente) funzione di G1 e G0;
- HF_C e F_C sono costanti dipendenti da F2, F1 e F0;
- V_{REF} è il riferimento di tensione e si utilizza quello interno al componente che vale 2.4 V.

Si riportano le tabelle di configurazione per le costanti HF_C e F_C:

F2	F1	F0	F_C (Hz), with MCLK = 3.58 MHz	HF_C (Hz), with MCLK = 3.58 MHz	HF_{OUT} (Hz), with full scale AC inputs
0	0	0	$MCLK/2^{21} = 1.71$	$64 \cdot F_C = 109.25$	23.71
0	0	1	$MCLK/2^{20} = 3.41$	$32 \cdot F_C = 109.25$	23.71
0	1	0	$MCLK/2^{19} = 6.83$	$16 \cdot F_C = 109.25$	23.71
0	1	1	$MCLK/2^{18} = 13.66$	$2048 \cdot F_C = 27968.75$	6070.12
1	0	0	$MCLK/2^{19} = 6.83$	$8 \cdot F_C = 54.62$	11.85
1	0	1	$MCLK/2^{22} = 0.85$	$64 \cdot F_C = 54.62$	11.85
1	1	0	$MCLK/2^{21} = 1.71$	$32 \cdot F_C = 54.62$	11.85
1	1	1	$MCLK/2^{20} = 3.41$	$16 \cdot F_C = 54.62$	11.85

Tabella 4.1: Configurazione delle costanti HF_C e F_C

e la tabella per la configurazione del guadagno G:

G1	G0	CH0 Gain	Maximum CH0 Voltage
0	0	1	±470 mV
0	1	2	±235 mV
1	0	8	±60 mV
1	1	16	±30 mV

Tabella 4.2: Configurazione della costante G

Il valore massimo per l'ingresso in tensione (CH1) è di 660 mV.
Le tabelle qui presentate verranno utilizzate in fase di taratura per la corretta impostazione delle funzionalità della scheda (cap. 6).

4.3.2 Il circuito d'ingresso

I due canali d'ingresso dell'integrato vengono connessi al carico attraverso i circuiti di figura 4.6 e 4.7.

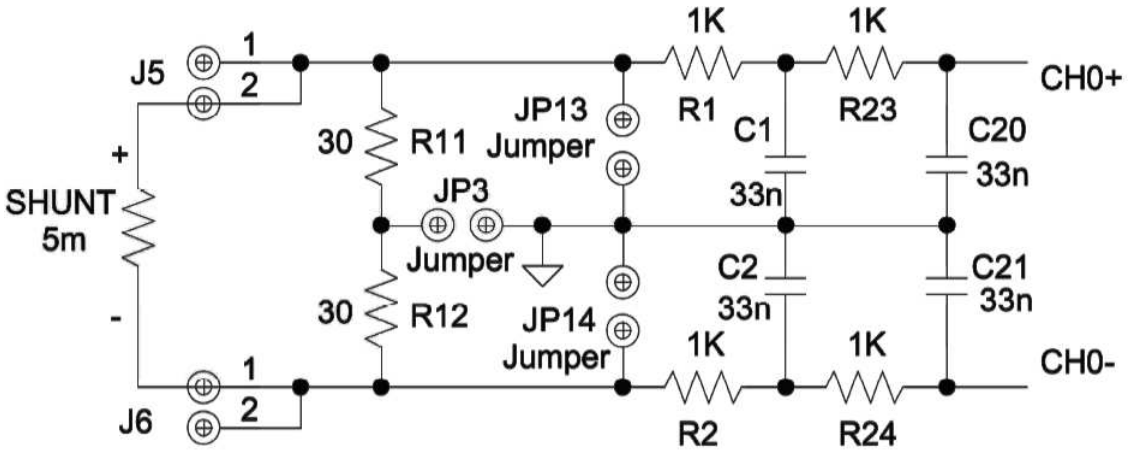

Figura 4.6: Circuito d'ingresso per la corrente

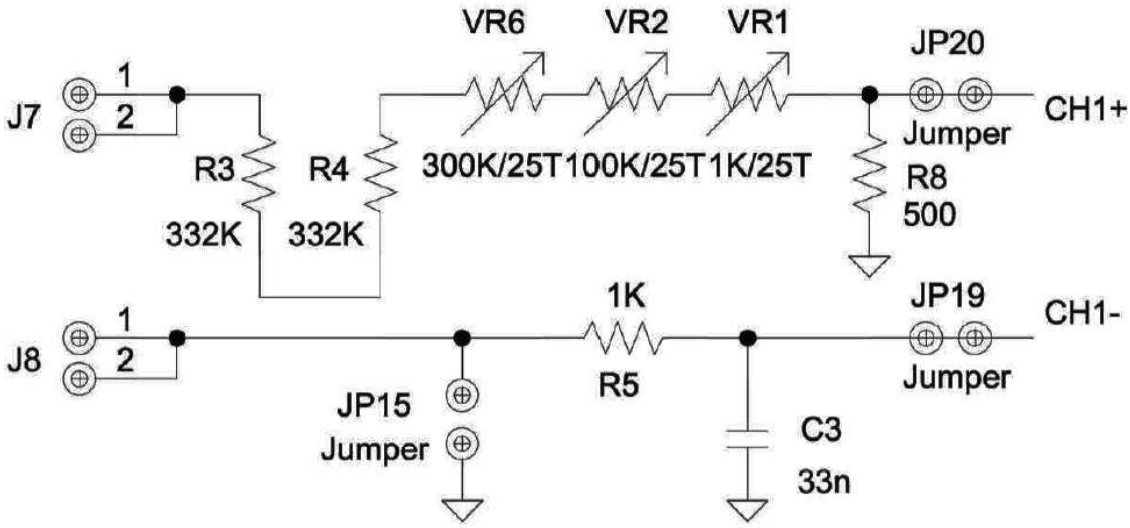

Figura 4.7: Circuito d'ingresso per la tensione

Nel circuito di figura 4.6 R11 e R12 vanno connesse nel caso si utilizzi un trasformatore TA al posto dello shunt. I due filtri del primo ordine in cascata su entrambi gli ingressi servono per la compensazione della parte induttiva dello shunt.

4.3.3 Il circuito d'uscita

La scheda è isolata galvanicamente dal microcontrollore grazie a un fotoaccoppiatore a quattro canali; in questo modo è garantito l'isolamento necessario, evitando che la tensione di rete possa interessare la parte digitale compromettendone la sicurezza.

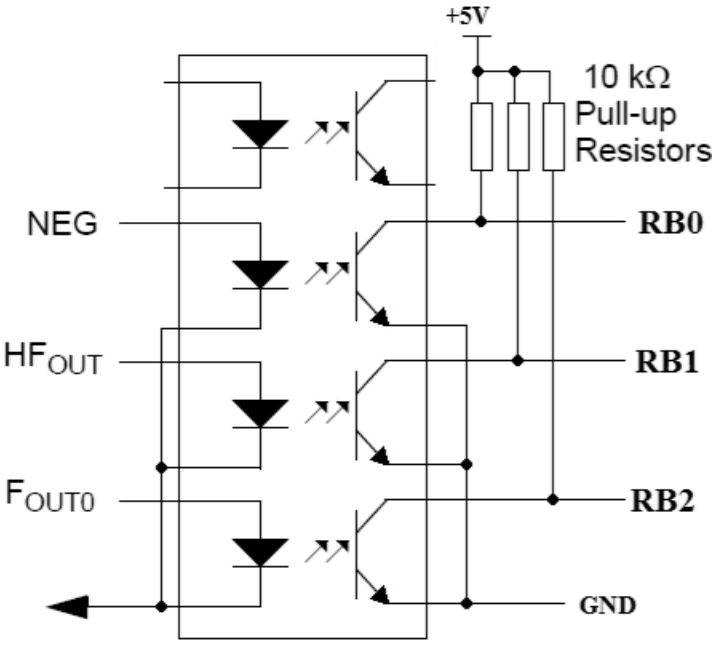

Figura 4.8: Circuito d'uscita

In figura 4.8 si vedono i tre segnali digitali: HF_{OUT}, F_{OUT_0} e NEG, che vengono interfacciati tramite il fotoaccoppiatore agli ingressi RB0, RB1 e RB2 del microcontrollore.

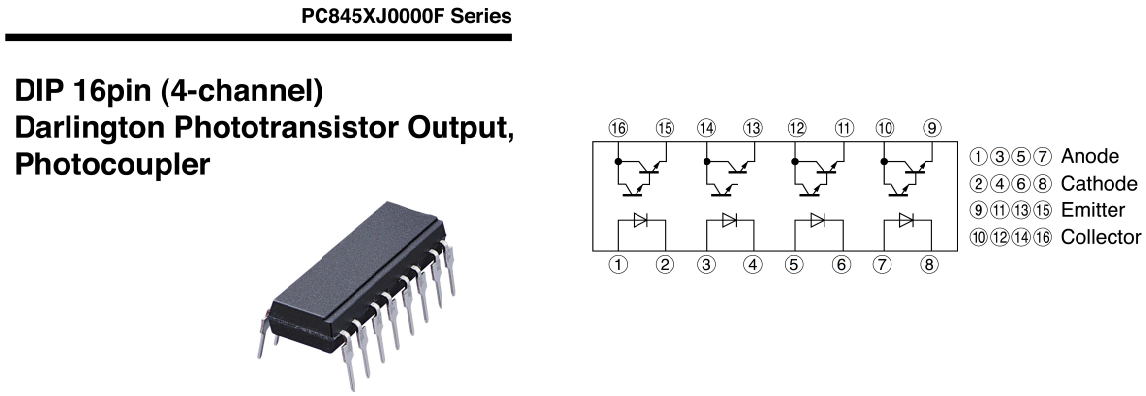

Figura 4.9: Fotoaccoppiatore a quattro canali PC845XJ0000F

4.4 Connessione alla rete del sensore

La scheda Energy Meter va connessa al carico e alla rete nel modo indicato dalla figura 6.8, l'effetto prodotto dall'inserimento della scheda Energy Meter è descritto dallo schema elettrico di figura 4.10:

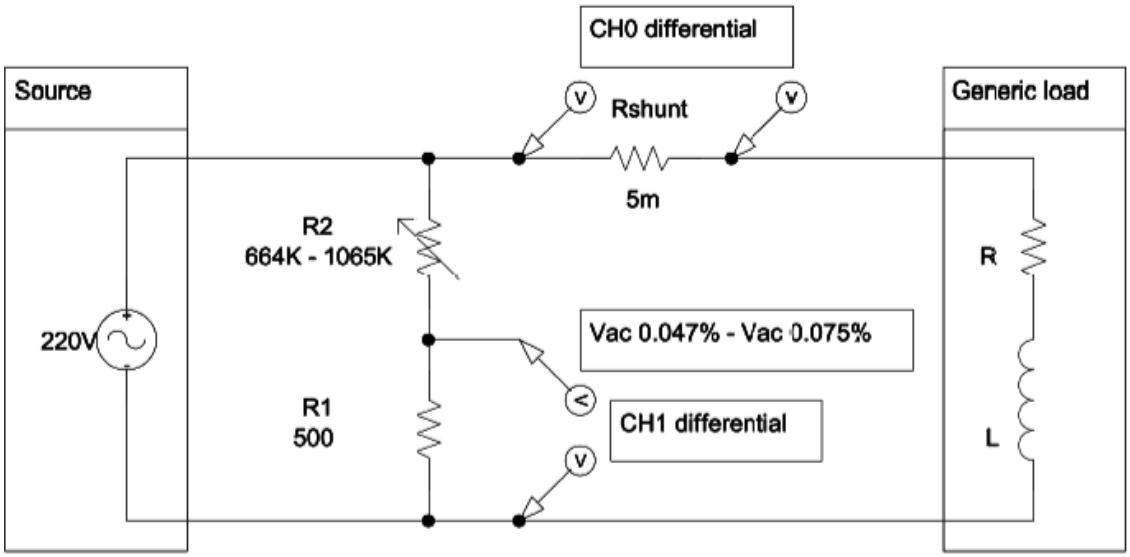

Figura 4.10: Schema elettrico di connessione alla rete del sensore

Dagli schemi d'ingresso 4.6 e 4.7 della scheda Energy Meter si nota che all'ingresso I arriva direttamente il valore di tensione ai capi dello shunt, mentre all'ingresso V è presente un partitore di tensione regolabile.
Il partitore in questione è formato da una serie di due resistori da 332 kΩ e di tre potenziometri di precisione (25 giri) di valori 300 kΩ, 100 kΩ e 1 kΩ; il ramo di bassa tensione è costituito da una resistenza di 500 Ω.
L'attenuazione del partitore ai due estremi della regolazione dei potenziometri, la cui resistenza totale in serie vale 401 kΩ, vale rispettivamente:

$$\alpha_{min} = \frac{500}{500 + (2 \cdot 332 \cdot 10^3)} = 752.4 \cdot 10^{-6} \tag{4.22}$$

$$\alpha_{MAX} = \frac{500}{500 + [(2 \cdot 332 \cdot 10^3) + 401 \cdot 10^3]} = 469.3 \cdot 10^{-6} \tag{4.23}$$

L'attenuazione della tensione d'ingresso è nell'intervallo $469.3 \cdot 10^{-6} < \alpha < 752.4 \cdot 10^{-6}$.

4.5 Acquisizione della frequenza

Per poter interpretare l'informazione proveniente dalla scheda Energy Meter MCP3905A Evaluation Board occorre avere un sistema che effettui misure di frequenza e di intervalli di tempo.
La soluzione proposta in questa tesi per eseguire una tale misurazione è visibile in figura 4.11 ed è spiegata di seguito: con l'arrivo di un fronte di salita all'ingresso RB1 del PIC, facente capo all'uscita HF_{OUT} dell'Energy Meter, comincia l'incremento di un'unità nella variabile *ms* a intervalli regolari di 1 ms (Start). La variabile *ms* viene incrementata fino all'arrivo del successivo fronte di salita dell'Energy Meter che ne arresta il conteggio[1]. In questa situazione (Stop) c'è la conversione di *ms* da millisecondi in Watt e la memorizzazione del dato convertito in un buffer.
In questo caso si è scelto di implementare un buffer circolare che permette di inviare all'NCAP, quando richiesto, un certo numero di misurazioni, che verranno visualizzate in un grafico direttamente tramite la porta seriale RS232.
Il buffer circolare ha dimensione definita dalla costante DIM_BUFFER nel file EnergyMeterRFD.c del WTIM; nel caso specifico abbiamo scelto di memorizzare dieci valori di potenza, oltre a quello dell'energia.
Si è scelto di implementare l'acquisizione in modo che il WTIM operi con cadenza non superiore ad una lettura al secondo, per evitare operazioni di lettura/scrittura sul buffer troppo frequenti.

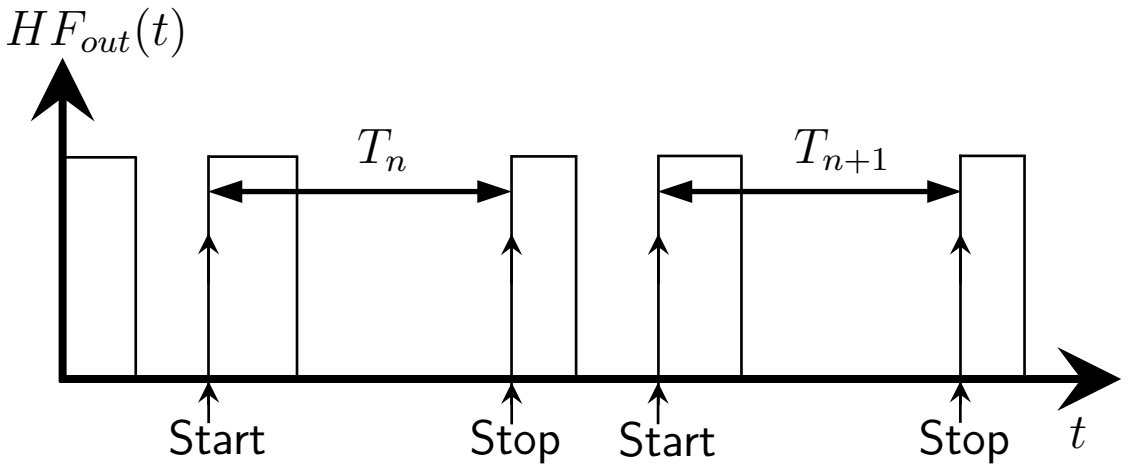

Figura 4.11: Principio di acquisizione della frequenza

[1]La variabile *ms* contiene la lettura dei millisecondi trascorsi (T_n) che è inversamente proporzionale alla potenza attiva.

Dallo schema di figura 4.11 si vede che gli istanti di Start e Stop non coincidono: in particolare se il periodo $T_n < 1$ s allora il prossimo periodo misurato T_{n+1} inizierà dopo che sia trascorso almeno 1 s dall'ultimo T_n misurato; al contrario se $T_n \geq 1$ s allora il successivo periodo misurato T_{n+1} sarà il nuovo fronte di salita che si verificherà subito dopo l'ultimo fronte di Stop. Il tempo minimo tra due letture è facilmente modificabile nella funzione *UserInterruptHandler(void)* nel PIC18 trattata nel cap. 5.
Grazie alla funzione di gestione dell'interrupt implementata riusciamo a gestire sia l'incremento automatico a opera di un timer interno, sia gli eventi di start e stop del conteggio dovuti ai fronti di salita sul pin RB1 del microcontrollore.
Il timer interno è impostato in modo da generare un interrupt ogni millisecondo, costituendo la base dei tempi.
A bordo della demoboard con il PIC18 c'è un quarzo da 4 MHz, grazie al PLL interno al PIC questa frequenza è moltiplicata di un fattore 4, ottenendo così un clock a 16 MHz; per avere la base dei tempi pari a un millisecondo si è fatto ricorso al prescaler interno del PIC.

Capitolo 5

Implementazione dell'applicazione software

In questo capitolo si intendono presentare le funzioni che sono state implementate per ottenere un sistema di misurazione wireless basato su letture di potenza ed energia. Le funzioni implementate riguardano le acquisizioni delle misure di frequenza e la comunicazione delle stesse tra WTIM e NCAP per mezzo delle funzioni IEEE 1451.

5.1 Funzioni implementate nel modulo NCAP (PIC24)

5.1.1 *UInt16 read(TimeDuration timeout, OctetArray *payload)*

La funzione *read* ha il compito di estrarre i dati dal PAYLOAD del pacchetto ZigBee e di interpretarli; infatti è noto che fino a questo punto ZigBee non sa che i dati ricevuti si riferiscono a letture di potenza ed energia. Nella seguente funzione tramite un ciclo *for* si riesce a sezionare la stringa di numeri in formato esadecimale del PAYLOAD nei punti corretti; i valori estratti vengono convertiti in numero e inseriti in un buffer di interi con segno, cioè nel formato giusto per essere manipolati.

Il numero massimo di valori che può essere trasferito tramite un singolo PAYLOAD è determinato dal valore della costante MAX_DIM_BUFFER che in questo caso è impostato a 15, valore sufficiente dato che il numero di valori trasmesso dal WTIM è di dieci letture di potenza più una di energia. Inoltre, la funzione stampa il primo valore che riguarda la lettura di energia e viene invocata la funzione *GraficaValori* spiegata di seguito per presentare i valori relativi alle letture di potenza.

```
1 #define MAX_DIM_BUFFER 15 //numero max di valori che può ricevere
2 static int valori[MAX_DIM_BUFFER];
3
4 UInt16 read(TimeDuration timeout, OctetArray *payload)
5 {
6  UInt8 lungDatoGetString;
7  unsigned int i,valore,indice2,indice3;
8  char valore;
9  char tempstring[] = "0123456789";  //10 cifre,
10 //stringa che sarà scomposta in valori per l'array.
11 payload->len = 0;
12 indice2 = 0; //contiene il numero di elementi estratti
13 indice3 = 0; //serve a misurare la lunghezza delle sottostringhe
14
15 //dopo il dato ci sono altri 2 caratteri,spazio e C
16 lungDatoGetString=APLGet();
17
18 //il ciclo viene bloccato con l'unità di misura 'W' o dal fine 0x00.
19 for(i = 0; (valore != 0x0)&&(valore != 'W'); i++)
20 {
21  valore = APLGet();
22  if(valore == 0x2e) //valore corrispondente al punto decimale
23  {
24    ConsolePutROMString( (ROM char *)"." );
25    i++;
26    payload->val[payload->len++]=(BYTE)".";
27  }
28  else if(valore == 0x20) //valore corrispondente allo spazio
29  {
30    ConsolePutROMString( (ROM char *)" " );
31    i++;
32    payload->val[payload->len++]=(BYTE)" ";
33  }
34  else if(valore == '-') //valore corrispondente al meno '-' oppure 0x5F
35  {
36    ConsolePutROMString( (ROM char *)"-" );
37    i++;
38    payload->val[payload->len++]=(BYTE)"-";
39    tempstring[indice3++] = (BYTE)'-';
40  }
41  else if(valore == 0x3b) //valore corrispondente al ;
42  {
```

```
43      ConsolePutROMString( (ROM char *)" " );
44      i++;
45      payload->val[payload->len++]=(BYTE)" ";
46      tempstring[indice3] = 0x0; //blocca qui la stringa
47      valori[indice2++]   = atoi(tempstring); //converti str to int
48      indice3 = 0x0;     //riparti
49    }
50    else if(valore == 0x43) //valore corrispondente al carattere maiuscolo C
51    {
52      ConsolePutROMString( (ROM char *)"C" );
53      i++;
54      payload->val[payload->len++]=(BYTE)"C";
55    }
56    else if(valore == 0x57) //valore corrispondente al carattere maiuscolo W
57    {
58      ConsolePutROMString( (ROM char *)"W" );
59      i++;
60      payload->val[payload->len++]=(BYTE)"W";
61    }
62    else
63    {
64      ConsolePut(valore);
65      payload->val[payload->len++] = valore - 0x30;
66      tempstring[indice3++] = (BYTE)valore;
67    }
68  }
69  // Nella stringa il primo valore ricevuto è l'energia:
70  ConsolePutROMString( (ROM char *)"\r\nContatore di energia: " );
71  myitoa( valori[0], tempstring );//converti uint -> string
72  ConsolePutROMString((char *)tempstring);
73  ConsolePutROMString( (ROM char *)" [W\\h]" );
74
75  indice2--; //riaggiusta il vettore
76  for(i=0;i<indice2;i++)
77   valori[i] = valori[i+1];
78
79  GraficaValori(valori,indice2);
80
81  return 0;
82 }
```

Elenco dei listati 5.1: read

5.1.2 *void GraficaValori(int x[], unsigned int dimarray)*

Come obiettivo principale la funzione si prepone la stampa dei valori passati in un array, in modo da creare un grafico sullo schermo del programma di comunicazione seriale. A questo si aggiungono la ricerca dei valori massimo e minimo e il calcolo della media tra tutti quelli contenuti nell'array. Tramite l'impostazione della costante AMPIEZZA_GRAFICO è possibile variare la scala della ampiezze del grafico.
Applicando questa funzione alle dieci letture di potenza estratte dal PAYLOAD si ottiene un'idea immediata dell'andamento dei consumi nel corso delle acquisizioni.

```
1  // funzione per graficare i valori RS232:
2  void GraficaValori( int x[], unsigned int dimarray)
3  {
4    #define AMPIEZZA_GRAFICO 20 //ampiezza verticale del grafico in caratteri
5    unsigned int i,j,max,min,media,pos;
6    char tempstr[] = "          ";
7    ConsolePutROMString( (ROM char *)"\r\nN valori acquisiti : " );
8    myitoa(dimarray, tempstr );    //converti uint -> string
9    ConsolePutROMString((char *)tempstr);
10   ConsolePutROMString( (ROM char *)"\r\n" );
11
12   for(i = 0; i < dimarray; i++) //stampa tutto il vettore acquisito..
13   {
14     myitoa( x[i], tempstr );//converti uint -> string
15     ConsolePutROMString((char *)tempstr);
16     ConsolePutROMString((ROM char *)" ");
17   }
18   j = 0;     //ora consideriamo il MODULO
19   for(i = 0; i < dimarray; i++) //stampa tutto il vettore acquisito..
20   {
21     if(x[i]<0)
22     {
23       x[i] *= -1;    //rendi positivo
24       if(j==0)
25       {
26         j = 1;         //solo una volta
27         ConsolePutROMString((ROM char *)"\r\nIl carico si comporta anche
               come bipolo attivo!\r\n");
28       }
29     }
30   }
31
32   max=x[0]; // trova il valore max, imposto il primo
```

```
33    for(i = 0; i < dimarray; i++)
34      if(x[i]>max) max = x[i];
35
36    min=x[0]; // trova il valore minimo, imposto il primo
37    for(i = 0; i < dimarray; i++)
38      if(x[i]<min) min = x[i];
39    //fai la media dei valori per sapere dove posizionarli
40    media = 0;  // imposto il primo
41    for(i = 0; i < dimarray; i++) //media = somma(Pi) / i
42      media += x[i];
43    media = media / dimarray;
44    //----------------------stampa dati----------------------
45    ConsolePutROMString( (ROM char *)"\r\nPotenza Attiva Istantanea:" );
46    ConsolePutROMString( (ROM char *)"\r\nMAX : " );
47    myitoa( max, tempstr );//converti uint -> string
48    ConsolePutROMString((char *)tempstr);
49    ConsolePutROMString( (ROM char *)" [W] " );
50    ConsolePutROMString( (ROM char *)"\t MIN : " );
51    myitoa( min, tempstr );//converti uint -> string
52    ConsolePutROMString((char *)tempstr);
53    ConsolePutROMString( (ROM char *)" [W] " );
54    ConsolePutROMString( (ROM char *)"\t Media : " );
55    myitoa( media, tempstr );//converti uint -> string
56    ConsolePutROMString((char *)tempstr);
57    ConsolePutROMString( (ROM char *)" [W]\r\n" );
58    //------------------------------------------------------
59    for(i = 0; i < dimarray*2; i++)
60      ConsolePut('-');
61    ConsolePutROMString( (ROM char *)"\r\n" );
62
63    //------ elaborazione dati e stampa ----------
64    //calcola la posizione in cui stampare il dato corrente
65    // posizione = [(valore - min) * AMPIEZZA_GRAFICO] / (max - min)
66    for(i=AMPIEZZA_GRAFICO;i>0;i--)
67    {
68      for(j=0;j<dimarray; j++)
69      {
70        pos = (((x[j] - min) * (AMPIEZZA_GRAFICO)) / (max - min + 1));
71        if((i-1) == pos)  ConsolePut('X');
72        else              ConsolePut(' ');
73        ConsolePut(' ');
74      }
75      ConsolePutROMString( (ROM char *)"\r\n" ); //a capo
```

```
76    }
77    for(i = 0; i < dimarray*2; i++)
78      ConsolePut('-');
79    ConsolePutROMString( (ROM char *)"\r\n" );
80  }
```

Elenco dei listati 5.2: GraficaValori

5.2 Funzioni implementate nel modulo WTIM (PIC18)

Occorre passare ora al WTIM e vedere come è definita l'acquisizione della frequenza tramite la funzione di interrupt:

5.2.1 *void UserInterruptHandler(void)*

La funzione deve gestire i tre tipi di interrupt utilizzati, dovuti ai seguenti eventi:

1. il cambio di livello logico sulla linea di interrupt esterno (ad alta priorità) proveniente dall'Energy Meter svolge la funzione di *start* e *stop* del contatore interno; in questo modo si gestisce in maniera automatica l'avvio e l'arresto del conteggio nella misurazione, all'interno di questa parte di codice c'è anche la conversione del dato da millisecondi in Watt, la gestione e memorizzazione del buffer circolare e l'acquisizione della lettura di energia.

2. l'overflow del contatore interno collegato al clock (diviso dal prescaler) è la base dei tempi;

3. il cambio di livello logico delle linee su PortB, in questo caso collegate ai 2 pulsanti a bordo della scheda.

```
1  void UserInterruptHandler(void)
2  {
3   unsigned long int potcounterlong3;
4   // verifica se l' interrupt è dovuto al fronte di discesa INT2
5   if (INTCON3bits.INT2IF)
6   {
7     INTCON3bits.INT2IF = 0;  // clear interrupt flag
8     if(flagenableacq == 1)
9     {
10      if(flagcounter==1)
11      {
```

```
12          flagcounter = 0;    // disabilita acquisizione
13          flagenableacq = 0; // esegui questa sezione una sola volta
14          //stampa i ms misurati
15          ConsolePutROMString((char *)"----------------\n\rPeriodo: ");
16          ultoa(potcounter,TxDataMS);  // converti int to string
17          ConsolePutString((char *)TxDataMS);  // invia il dato letto - # ms
18          ConsolePutROMString((char *)" [ms]\n\rPotenza attiva ed energia: ");
19          //aggiungi unità di misura - [ms]
20          //potcouner ha il periodo della frequenza da misurare.
21          Frequenzimetro(potcounter); //stampa periodo e frequenza
22
23          // Conversione in Watt con la proporzione
24          // 4.4 KW 500ms = (x)_watt (counter)_ms
25          wattcounter = CONST_CONVERSIONE_MS_TO_WATT / potcounter;
26          // Memorizza nel buffer circolare grande DIM_BUFFER
27          if(Neg==0) wattcounter *= (-1); // pot att negativa
28          //se ho riempito tutto riparto
29          if(potistbufferindex>=DIM_BUFFER) potistbufferindex = 0;
30          potistbuffer[potistbufferindex++]=wattcounter;
31          ltoa(wattcounter,TxDataW);
32          ConsolePutString((char *)TxDataW);
33          ConsolePutString((char *)StringWatt);  // unità di misura- # W
34          if(Fout == 0 && flagEnergy == 1) //----Contatore di energia----
35          {  // aspetta il prossimo fronte per incrementare ancora
36              flagEnergy = 0;
37              energycounter++;
38          }
39          else if(Fout == 1)// riabilita l' incremento al prox fronte
40              flagEnergy = 1;
41          // Leggi il segnale a bassa frequenza - energia
42          ultoa(energycounter,TxDataEnergy);
43          ConsolePutString((char *)TxDataEnergy);
44          ConsolePutString((char *)StringEnergy);  // # Wh
45          //stampa tutto il contenuto del buffer
46          ConsolePutROMString( (ROM char *)"\r\n" );
47          for(indice1 = 0; indice1 < DIM_BUFFER; indice1++)
48          {
49              ltoa(potistbuffer[indice1],TxdataOn);
50              ConsolePutString((char *)TxdataOn);
51              ConsolePutROMString( (ROM char *)" ");  // spazio
52          }
53          ConsolePutROMString( (ROM char *)"\r\n" );
54          potcounter = 0;  // reset contatore
```

```
55        }
56        else
57            flagcounter = 1;  // abilita acquisizione
58      }
59    }
```

Elenco dei listati 5.3: UserInterruptHandler

Si passa poi alla seconda causa di interrupt, verificando se l'interrupt è dovuto al timer sincronizzato con il clock diviso dal prescaler. Questa parte è la nostra base dei tempi:

```
1   if(PIR1bits.TMR2IF == 1) //esegue ogni 1 ms //
2   {
3     PIR1bits.TMR2IF =0;     // resetta il flag
4     ms++;  // incrementa i millisecondi con l'attivazione del timer
5     if(flagcounter==1&&flagenableacq == 1)
6     {
7         potcounter++;  // se sono in fase di acquisizione del periodo
8     }                  // incremento il counter dei ms
9     if(ms==1000)      // ogni 1 s //
10    {
11        ms=0;
12        six_sec++;       // incrementa i secondi trascorsi
13        flagenableacq = 1;  // ogni secondo abilita acquisizione
14    }  // free running
15    if(six_sec==60)  //esegue ogni minuto = 60 s
16        six_sec=0;
17  }
```

Infine si verifica se l'interrupt sia causato dal cambiamento di stato degli switch:

```
1   if(INTCONbits.RBIF == 1)    //Check for Change Notification interrupt
2   {
3      INTCONbits.RBIF = 0;     //Clear Change Notification interrupt
4      INTCONbits.RBIE = 0;     //disabilita interrupt
5      flag = 1;
6      if(Puls0==1)  LED0 = 1; //leggi i pulsanti
7      else  LED0 = 0;
8      if(Puls1==1)  LED1 = 1;
9      else  LED1 = 0;
10  }
11 }
```

5.2.2 *BYTE GetPowerMeterString(char *buffer)*

La seguente funzione si occupa di inserire nel buffer di trasmissione il PAYLOAD, contenente la lettura di energia e le letture di potenza attiva istantanea nel buffer:

```
1  BYTE GetPowerMeterString(char *buffer)
2  {
3   char  *ptr;
4   BYTE  strLen;
5   char  i,k;
6
7   ptr = buffer;
8
9   //invia la misura di energia:
10  ltoa(energycounter,TxdataOn); // converti long to string
11  strLen = strlen( TxdataOn );
12  for( i=0; i < strLen ; i++)
13      *ptr++ = TxdataOn[i];
14
15  *ptr++ = ';';  //separatore tra un dato ed il successivo ';'
16  //invia il buffer con le misure di potenza attiva:
17  for(indice1 = 0; indice1 < DIM_BUFFER; indice1++)
18  {
19    // Se il numero è negativo scrivi il segno meno
20    if(potistbuffer[indice1] == '-')
21      *ptr++ = '-';
22
23    ltoa(potistbuffer[indice1],TxdataOn); // converti long to string
24    strLen = strlen( TxdataOn );
25    for( i=0; i < strLen ; i++)
26        *ptr++ = TxdataOn[i];
27
28    *ptr++ = ';';  //separatore tra un dato ed il successivo ';'
29  }
30  *ptr++ = ' ';  // spazio
31  *ptr++ = 'W';  // scrivi unità di misura, indica la fine dei dati
32  *ptr++ = '\n'; // a capo
33  *ptr   = '\0'; // fine stringa
34  return strlen( buffer );
35 }
```

Elenco dei listati 5.4: GetPowerMeterString

5.2.3 *void Frequenzimetro(unsigned long int periodo)*

Questa funzione potrà essere omessa nei futuri sviluppi del progetto per risparmiare memoria, in quanto serve solo come verifica del corretto funzionamento dell'acquisizione di frequenza; in particolare la funzione riceve un valore intero e si occupa di stamparlo su RS232 con la giusta unità di misura. La funzione converte anche il valore intero in frequenza tramite la relazione $F = T^{-1}$ gestendo l'eventuale overflow che può crearsi nella divisione e lo stampa su RS232.

```
1  //La funzione stampa frequenza e periodo, range: da 1ms a 60s
2  void Frequenzimetro(unsigned long int periodo)
3  {
4   unsigned long int contatorepot;
5   ultoa(periodo,TxDataMS); // converti int to string
6   ConsolePutString((char *)TxEnter); // a capo
7   // se è minore di zero dividi per KiloWatt
8   if(TxDataMS[0]=='-') // evita overflow e visualizza decimali
9   {
10    contatorepot = periodo / 1000;
11    ultoa(contatorepot,TxDataMS); // converti long to string
12    ConsolePutString((char *)TxDataMS); // invia il dato letto
13    ConsolePutROMString((char *)" [s]\t "); // [s]
14  }
15  else
16  {
17    ConsolePutString((char *)TxDataMS); //  invia il dato letto
18    ConsolePutROMString((char *)" [ms]\t "); // [ms]
19  }
20  // conversione T [ms] -> F [Hz]
21  contatorepot = 1000 / periodo; // estraggo parte intera
22  ultoa(contatorepot,TxdataOn);  // converti long to string
23  ConsolePutString((char *)TxdataOn);
24  ConsolePutString((char *)TxDot);      //tx punto
25  contatorepot  = contatorepot * 1000; // parte intera * 1000
26  contatorepot = (1000000 / periodo) - contatorepot;
27  // resto = (numero * 1000 con tutti i decimali)- parte intera
28  ultoa(contatorepot,TxdataOn); // converti long to string
29  ConsolePutString((char *)TxdataOn); // invia il dato letto
30  ConsolePutROMString( (ROM char *)" [Hz]\t\r\n" ); // Hz
31 }
```

Elenco dei listati 5.5: Frequenzimetro

5.3 L'applicazione software

L'applicazione per essere testata richiede che siano utilizzati insieme i due sistemi che prendono il nome di NCAP e WTIM. Nello specifico si utilizza una demoboard EXPLORER 16 come nodo NCAP e una demoboard PICDEM™Z come nodo WTIM. Entrambi i moduli hanno a bordo un'antenna e transceiver MRF24J40 e un collegamento seriale RS232 che permette di visualizzare i messaggi provenienti dai nodi e di inviare dei comandi; una volta configurata e aperta la comunicazione seriale con i parametri 19200,8,N,1 è possibile alimentare la board.
All'avvio, dopo i processi preliminari di inizializzazione dell'hardware, l'NCAP crea una rete (figura 5.1) e manda a video un menu tramite il quale l'utente può inizializzare la rete per sfruttare le funzionalità dello standard IEEE 1451. Da questo momento in poi il WTIM può collegarsi autonomamente alla rete: l'NCAP lo riconosce e lo registra inserendolo nella lista dei TIM Dot5. Dopo aver dato notifica dell'avvenuta registrazione, l'NCAP visualizza un menu con le opzioni di comunicazioni specifiche per quel TIM.
A questo punto l'NCAP fornisce le liste di moduli, TIM e canali presenti da cui l'utente può selezionare il dispositivo desiderato; in questa fase l'utente, scegliendo prima il modulo, poi il TIM e successivamente il canale non fa altro che chiamare le funzioni reportCommModule, reportTims e reportChannels, impostandone i parametri di ingresso in funzione delle scelte fatte.

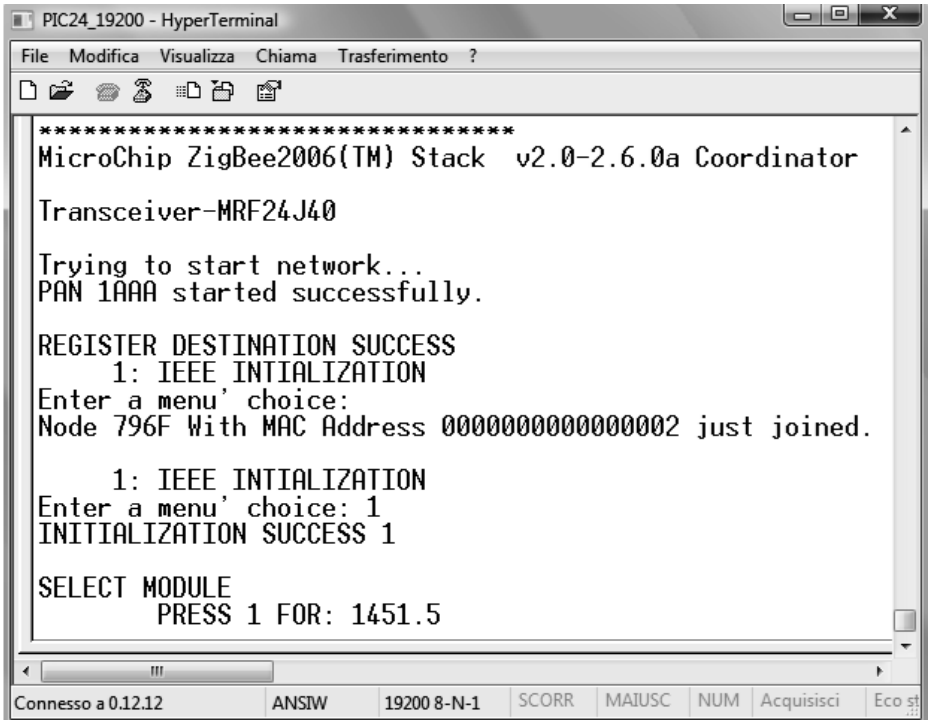

Figura 5.1: Messaggi PIC24 - inizializzazione

Lo scambio di messaggi necessario per la ricezione del dato di misura inizia con la sequenza di operazioni qui descritta.

- NCAP esegue open(), che ritorna il transCommId e imposta i parametri di connessione modificando i campi della primitiva APSDE_DATA_request;

- la funzione readData() compone il Command Message che potrebbe contenere la richiesta al WTIM di fornire le misure, e lo trasferisce al payload IEEE 1451.

  L'NCAP chiama la funzione write() che carica il buffer di trasmissione di ZigBee con il payload IEEE 1451 ed effettua la trasmissione al WTIM, trasferendo al ciclo macchina interno la primitiva ZigBee APSDE_DATA_request.

- Il WTIM riceve il messaggio e, al livello del protocollo ZigBee genera la primitiva APSDE_DATA_indication che contiene le informazioni sul mittente del messaggio. La funzione ProcessZigBeePrimitive del WTIM analizza i campi di questa primitiva e ne estrae il clusterID, mentre il WTIM scompatta il messaggio ricevuto in base alla descrizione fornita dal cluster e lo decodifica.

  Quando il messaggio è una richiesta dei dati di misura di potenza ed energia il WTIM effettua una copia del buffer nel PAYLOAD e compone il Reply Message, caricando tutti i rispettivi campi nel suo buffer di trasmissione con la funzione *BYTE GetPowerMeterString(char *buffer).*
  A questo punto il WTIM imposta la sua primitiva APSDE_DATA_request e la esegue, inviando quindi il messaggio di risposta all'NCAP tramite ZigBee.

- Lo strato ZigBee di NCAP riceve il messaggio e genera la primitiva APSDE_DATA_indication che contiene le informazioni sul mittente del messaggio. La funzione ProcessZigBeePrimitive dell'NCAP analizza i campi di questa primitiva e ne estrae il clusterID.
  Grazie alla funzione *UInt16 read(TimeDuration timeout, OctetArray *payload)* l'NCAP scompatta il messaggio ricevuto in base alla descrizione fornita dal cluster e recupera i dati di potenza ed energia, che vengono formattati in un array di interi e trasferiti allo strato IEEE 1451.5 e IEEE 1451.0.

- Con la funzione *void GraficaValori(int x[], unsigned int dimarray)* l'NCAP invia al collegamento seriale RS232 i valori di potenza ed energia (figura 5.2) inserendo in un grafico tutte le letture di potenza presenti nel buffer e calcolandone il minimo, il massimo e la media.

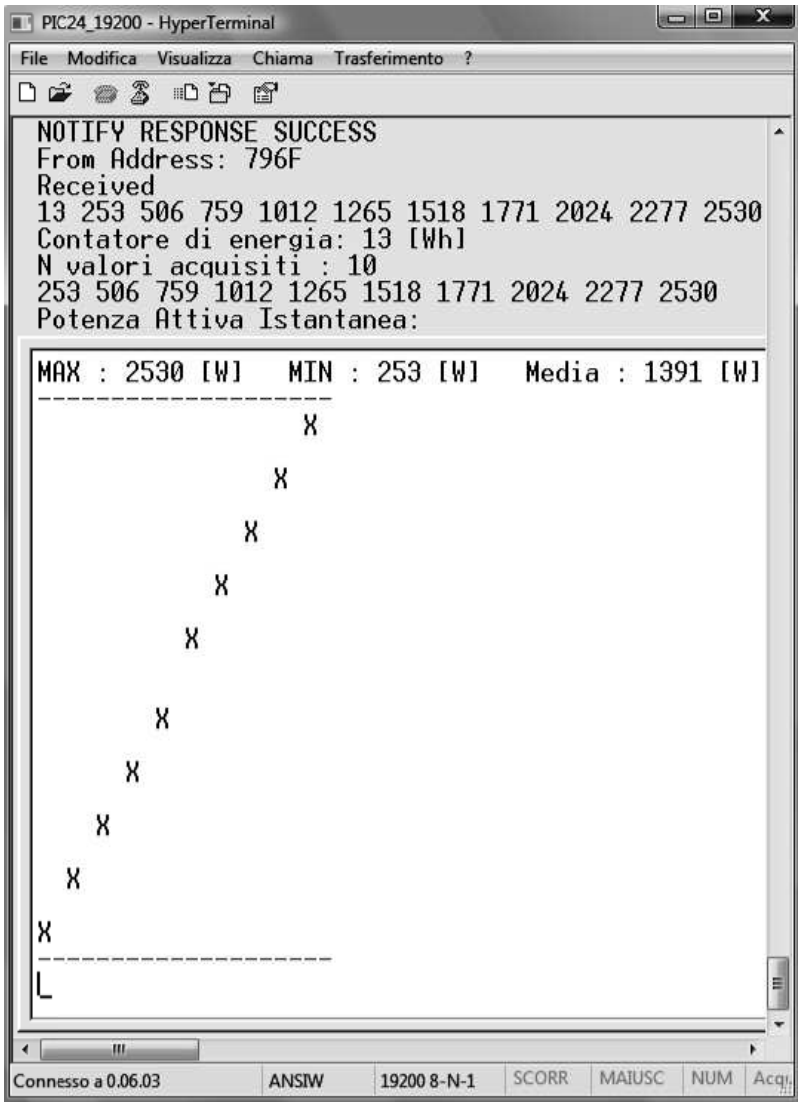

Figura 5.2: Messaggi PIC24 - ricezione buffer di valori

Dalla figura 5.2 è possibile leggere il report della sequenza appena descritta; in particolare si nota che dopo *received* c'è una linea di numeri che fanno parte del PAYLOAD e quindi in questo punto ZigBee ha ricevuto i dati, pur non sapendo ancora che si tratta di valori di potenza. Nel passaggio successivo con la funzione *read()* la stringa viene interpretata e di conseguenza vengono estratte le letture di energia e tutti i valori di potenza. Questi ultimi vengono inseriti in un buffer di tipo intero con segno e sono a disposizione dell'applicazione. Il numero 10 in figura 5.2 si riferisce al numero di valori delle letture di potenza ricevuti.

Si ricorda infine che il segno della potenza si riferisce alla direzione della potenza attiva verso il carico, cioè se positivo è entrante, altrimenti è uscente.

Si veda ora la sequenza delle varie fasi nel WTIM: come si può notare (figura 5.3), dopo l'accensione del nodo WTIM questo cerca una rete a cui connettersi e tenta di registrarsi con un indirizzo che attualmente è pre-assegnato in fase di compilazione (1AAA).
Dopo l'avvenuto successo nella creazione della rete si ha la registrazione con il protocollo IEEE 1451 (NCAPREG STATE) e il relativo successo (REGISTRATION: SUCCESS).
Dopo questa fase preliminare in cui il WTIM è entrato a far parte di una rete IEEE 1451 si vede il risultato dell'acquisizione di un periodo e la lettura di energia(Periodo: 500 [ms] ecc..) e la conversione in potenza attiva (Potenza attiva ed energia: 4400 [W] 13 [Wh]).
In questo caso dimostrativo la potenza misurata corrisponde al fondo scala dell'intero sistema pari a 4.4 kW[1] ed è stata misurata un'energia già assorbita di 13 Wh.
Viene anche aggiornato e ristampato di volta in volta il buffer circolare con tutti i valori di potenza attiva (Buffer [W]: 4400...).

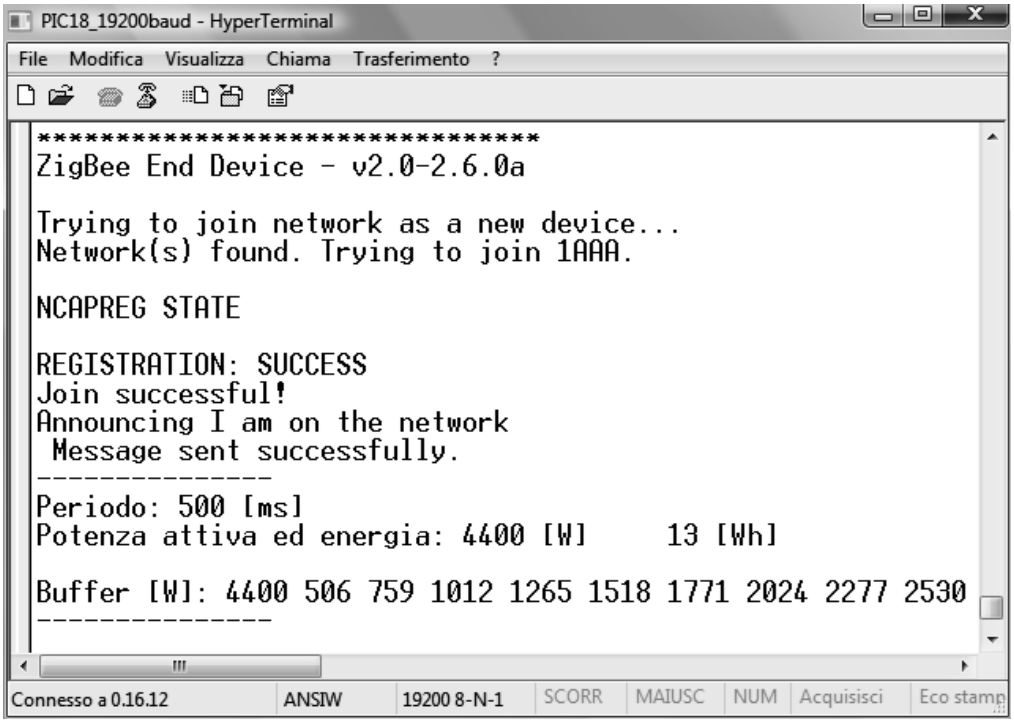

Figura 5.3: Messaggi PIC18

[1]Il valore di fondo scala è ottenuto grazie a un'emulazione del sistema tramite un generatore di segnale, questi test saranno spiegati in modo approfondito nel cap. 6.

Capitolo 6

Taratura e misure

6.1 Taratura del sensore di energia

Per tarare il sensore bisogna preliminarmente scegliere alcuni parametri. Nell'ipotesi di un'utenza domestica si presenta una tensione di lavoro di $220\,\mathrm{V_{rms}} \pm 10\%$.
I dati dello **shunt** a disposizione sono:

- $I_{MAX} = 20\,\mathrm{A}$
- $R_{shunt} = 5\,\mathrm{m\Omega}$
- $V_{MAX} = 100\,\mathrm{mV} \quad @ \quad I_{MAX}$
- Classe = 0.5 (tolleranza dello 0.5%)

6.1.1 Potenza a fondo scala

Con lo shunt a disposizione possiamo raggiungere una potenza di:

$$P_{MAX} = I_{rms_{MAX}} \cdot V_{rms_{MAX}} = 4.4 \pm 10\% \quad [kW] \tag{6.1}$$

Dal datasheet dell'integrato MCP3905A si sa che a fondo scala i valori massimi di tensione agli ingressi I e V sono rispettivamente:

$$V_{CH0_{MAX}} = 470/Gain \quad [mV] \tag{6.2}$$

$$V_{CH1_{MAX}} = 660 \quad [mV] \tag{6.3}$$

dove $Gain = 1$ con le impostazioni correnti scelte per i jumper.
Si riprende la formula 4.21, ossia:

$$HF_{OUT} = \frac{8.06 \quad V_0 \quad V_1 \quad G \quad HF_C}{(V_{REF})^2} \quad [Hz] \tag{6.4}$$

In teoria con i valori massimi di tensione appena citati è possibile ottenere un valore massimo di frequenza a fondo scala pari a:

$$HF_{OUT_{MAX}} \quad \cong \quad 23.71 \quad [Hz] \tag{6.5}$$

cioè un segnale con periodo:

$$T_{min} \quad \cong \quad 42.2 \quad [ms] \tag{6.6}$$

con le impostazioni dei jumper riassunte in tabella 6.1.
Questo dato è molto importante perché fissa l'ordine di grandezza minimo per avere una sufficiente precisione nella misura.
Per completare il discorso si aggiunga anche l'informazione di energia: l'MCP3905A fornisce infatti anche un'altra uscita F_{OUT} che rappresenta la misura di energia e che consiste in un impulso rettangolare che incrementa un contatore nel WTIM. Si è scelto di poter leggere anche questo segnale grazie al quadruplo optoisolatore aggiunto. Il valore di F_{OUT} è proporzionale a HF_{OUT} e nel nostro caso, con le impostazioni scelte per i jumper, il fattore di proporzionalità vale 64, quindi a fondo scala si avrà:

$$F_{OUT_{MAX}} \quad \cong \quad 0.37 \quad [Hz] \tag{6.7}$$

cioè un segnale con periodo:

$$T_{min} \quad \cong \quad 2.7 \quad [s] \tag{6.8}$$

6.1.2 Impostazione dei jumper sulla scheda

L'impostazione dei jumper è indicata nella tabella 6.1. Con la configurazione scelta si ottengono i seguenti valori: $V_{REF} = 2.4$ V, $MCLK = 3.58$ MHz, $HF_C = 109.25$, $G = 1$.
Ci si propone di trovare il periodo del segnale corrispondente alla massima potenza; sapendo che $V_{0_{MAX}} = V_{Shunt_{MAX}} = 100\,\text{mV}$ e sostituendo tutti i dati:

$$HF_{OUT_{MAX}} \quad = \quad V_1 \cdot 15.2874 \quad [Hz] \tag{6.9}$$

scrivendo $V_1 = \alpha \cdot V_{rms}$, si trova:

$$HF_{OUT_{MAX}} \quad = \quad V_{rms} \cdot \alpha \cdot 15.2874 \quad [Hz] \tag{6.10}$$

Grazie alla regolazione dei potenziometri si avrà un'escursione tra i seguenti valori:

$$HF_{OUT_{MAX}} = \begin{cases} V_{rms} \cdot \alpha_{min} \cdot 15.2874 \\ V_{rms} \cdot \alpha_{MAX} \cdot 15.2874 \end{cases} \cong \begin{cases} 2.53 \\ 1.58 \end{cases} \quad [Hz] \tag{6.11}$$

che corrispondono ai seguenti valori di periodo:

$$T_{MAX} \quad \cong \quad 395.3 \div 632.9 \quad [ms] \tag{6.12}$$

Jumper	Valore	Nome	Note
JP0	0	G0	Config. Guadagno CH0
JP1	0	G1	Config. Guadagno CH0
JP3	N.C.		resistenza tra CH0 e massa
JP4	+5 V	PWR	Alimentazione esterna
JP5	N.C.	OPT	Uscita segnale NEG
JP6	Connesso	LED	NEG su Led
JP12	1	HPF	Filtro Passa Alto attivo
JP8	0	F2	Config. Funzione Out
JP2	0	F1	Config. Funzione Out
JP11	0	F0	Config. Funzione Out
JP13	N.C.		CH0+ a massa
JP14	N.C.		CH0- a massa
JP15	N.C.		CH1- a massa
JP18	N.C.	CH1+	In V dell'IC
JP19	Connesso	CH1-	In V dell'IC
JP20	Connesso	CH1+	In V dell'IC
JP21	N.C.	CH1-	In V dell'IC

Tabella 6.1: Configurazione dei jumper

6.1.3 Coefficiente di proporzionalità

Secondo le impostazioni iniziali scelte è utile definire il coefficiente K, che permette di trasformare la lettura in millisecondi X_{ms} in potenza attiva impostando la seguente proporzione:

$$P_{misurata} \cdot X_{ms} \quad = \quad P_{MAX} \cdot T_{MAX} \quad = \quad \mathsf{K} \quad [Wms] \tag{6.13}$$

Scelgo un valore intermedio tra quelli calcolati, assumendo che $T_{MAX} = 500\,\text{ms}$ e con $P_{MAX} = 4.4\,\text{kW}$ calcolato inizialmente ottengo:

$$\mathsf{K} \quad = \quad P_{MAX} \cdot T_{MAX} \quad = \quad 2.2 \cdot 10^6 \quad [Wms] \tag{6.14}$$

da cui si ricava la lettura di potenza attiva con:

$$P_{misurata} \quad = \quad \frac{\mathsf{K}}{X_{ms}} \quad [W] \tag{6.15}$$

6.1.4 Minima potenza misurabile

Con i dati a disposizione della scheda Energy Meter è noto che la minima potenza rilevabile dall'integrato MCP3905A è pari a:

$$P_{min} \quad = \quad P_{FS} \cdot 0.0015\% \tag{6.16}$$

e nel nostro caso conviene far riferimento alla minima frequenza possibile cioè:

$$HF_{OUT_{min}} = HF_{OUT_{MAX}} \cdot 0.0015\% \cong 336 \quad [\mu Hz] \tag{6.17}$$

Per dare un'idea dell'ordine di grandezza con il coefficiente di proporzionalità in formula 6.14 questo valore può corrispondere a:

$$P_{min} \quad \cong \quad 780 \quad [mW] \tag{6.18}$$

Scendendo al di sotto di tale valore l'integrato non fornisce più alcun impulso.

6.2 Accuratezza delle misure

In questa sezione vengono presentate le misure effettuate sul WTIM con l'ausilio di un oscilloscopio e di due generatori di segnali per valutare l'accuratezza delle indicazioni che contengono l'informazione sulla potenza attiva utilizzata.

6.2.1 Il WTIM come frequenzimetro

Il grafico di fig. 6.2 illustra l'errore sull'acquisizione della frequenza del WTIM confrontato con il valore letto da un oscilloscopio: a entrambi gli ingressi degli strumenti è collegato un generatore di segnali che simula l'onda rettangolare del sensore di energia secondo lo schema di fig. 6.1.

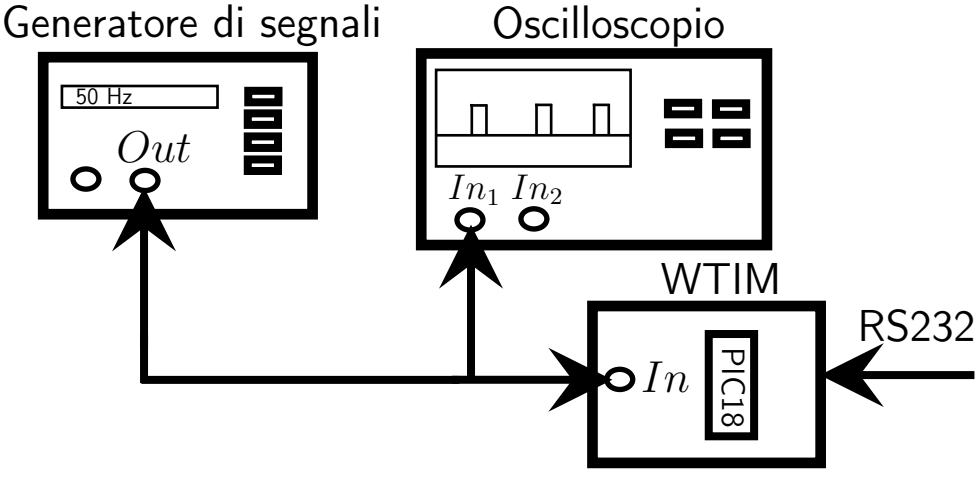

Figura 6.1: Schema connessione come frequenzimetro

Di seguito presento il grafico delle misure di periodo eseguite dal WTIM con diversi valori di frequenza del generatore di segnali:

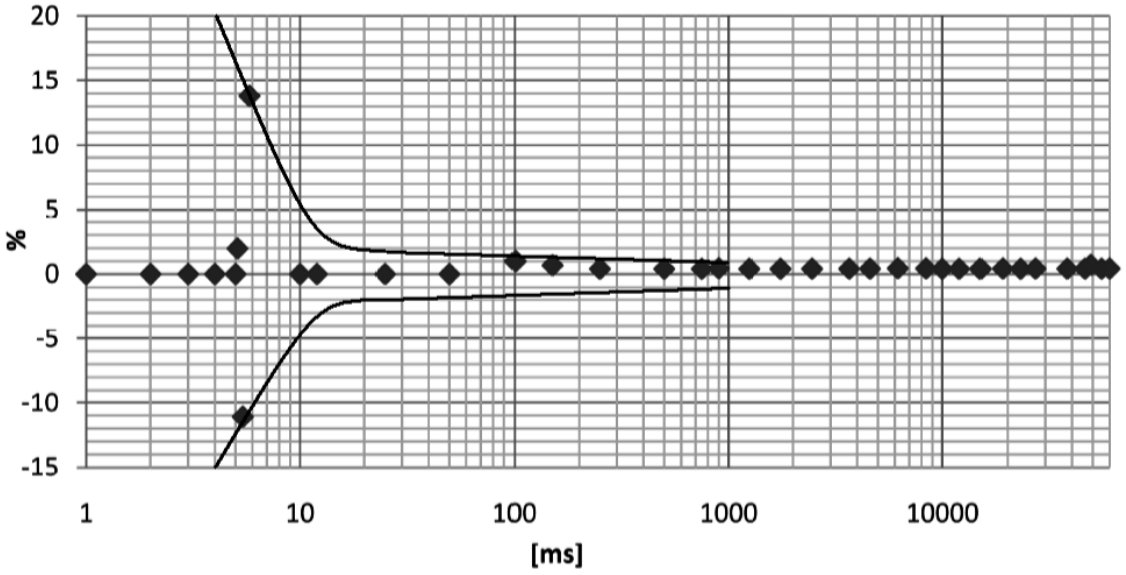

Figura 6.2: Accuratezza misura periodo

In prima analisi si ipotizza un'incertezza di $\pm$ 1 ms perché, confrontando la misura con la base dei tempi a 1 ms, è questo il valore di risoluzione che il PIC raggiunge nella misura. Si può avere infatti la sovrapposizione del fronte di clock e del fronte del segnale esterno sia all'apertura, sia alla chiusura del conteggio e questo causa una errata interpretazione della lettura.

Attorno a valori dell'ordine di qualche *ms*, basta un piccolo scostamento di qualche frazione di *ms* del segnale da misurare per rilevare il fronte errato, o il successivo o il precedente; questo è facilmente visibile nella misurazione eseguita e riportata di seguito. Ricordo inoltre che il WTIM mantiene solo valori interi a partire da 1 ms perciò con valori non interi prossimi a questo l'errore sarà al massimo di 1 ms.

A prova di ciò è evidente dal grafico 6.2 che, avvicinandosi a valori prossimi a 1 ms (cioè le righe della tabella 6.2 comprese tra i numeri cinque e otto), l'errore disegna una linea che tende asintoticamente al 100% per valori ancora più piccoli, come previsto.

Spostandosi invece verso valori dell'ordine di 10 s si nota che il WTIM tende ad acquisire un valore più elevato rispetto all'oscilloscopio; dal grafico 6.2 si vede che l'errore in percentuale rimane contenuto.

I motivi di tale comportamento sono principalmente due: il primo è la precisione del clock, perché sappiamo che il PIC utilizza un oscillatore con quarzo e per tempi sempre più grandi l'errore dovuto alla tolleranza del valore del quarzo è sempre più rilevante.

n	$T_{Oscilloscopio}$ [ms]	T_{WTIM} [ms]	err $(T_{Osc} - T_{WTIM})$ [ms]	$err\%$ $(100 \cdot err/T_{Osc})$ %
1	1	1	0	0
2	2	2	0	0
3	3	3	0	0
4	4	4	0	0
5	5	5	0	0
6	5.1	5	0.1	1.96
7	5.4	6	-0.6	-11.11
8	5.8	5	0.8	13.79
9	10	10	0	0
10	12	12	0	0
11	25	25	0	0
12	50	50	0	0
13	101	100	1	0.99
14	150	149	1	0.66
15	250	249	1	0.4
16	500	498	2	0.4
17	750	747	3	0.4
18	900	896	4	0.44
19	1250	1245	5	0.4
20	1750	1743	7	0.4
21	2460	2450	10	0.41
22	3670	3655	15	0.41
23	4590	4571	19	0.41
24	6200	6173	27	0.43
25	8400	8364	36	0.43
26	10000	9960	40	0.4
27	12000	11952	48	0.4
28	15000	14940	60	0.4
29	19000	18920	80	0.42
30	23000	22907	93	0.4
31	27000	26887	113	0.42
32	38000	37844	156	0.41
33	46000	45813	187	0.4
34	49000	48659	341	0.69
35	55000	54776	224	0.4
36	60000	59751	249	0.41

Tabella 6.2: Misure - WTIM come frequenzimetro

L'altra causa è la possibile interferenza delle temporizzazioni dello stack ZigBee durante le fasi di acquisizione di periodi dell'ordine delle decine di secondi, dall'implementazione delle funzioni è noto però che l'interrupt dell'acquisizione è impostato su priorità massima, in modo che la misura del periodo non venga disturbata dalle altre operazioni che compie il PIC, in questo caso la gestione dello stack ZigBee.
Dopo questa prima serie di misure è possibile affermare che il WTIM mantiene una accuratezza sufficiente nel range richiesto, mediamente pari a circa 0.4%, e da calcoli precedenti si sa che questo range è di $\approx 40\,\text{ms} \div 40\,\text{s}$.

6.2.2 Emulazione di misure di potenza

Utilizzando le informazioni appena trovate si sostituisce ora il generatore di segnale con il sensore di energia al quale, per ottenere misure più precise, non si è messa in ingresso la tensione di rete con lo shunt e un carico, ma due generatori di segnali sincronizzati con sfasamento nullo, secondo lo schema di fig. 6.3.
Nell'effettuare le misurazioni si è imposto un valore fisso come riferimento di tensione sul generatore di segnali n°2 pari a $2.2\,\text{V}_\text{P} = 2.674\,\text{V}_\text{rms}$ che simula sul canale CH1 la tensione di rete a 220 V, attenuata dal partitore d'ingresso. La sincronizzazione attuata fa si che lo sfasamento tra i segnali dei due generatori sia nullo e le misure differiscano per le sole variazioni del valore di tensione applicato all'ingresso di corrente CH0 ad opera del generatore di segnali n°1, simulando così carichi di potenza diversa.

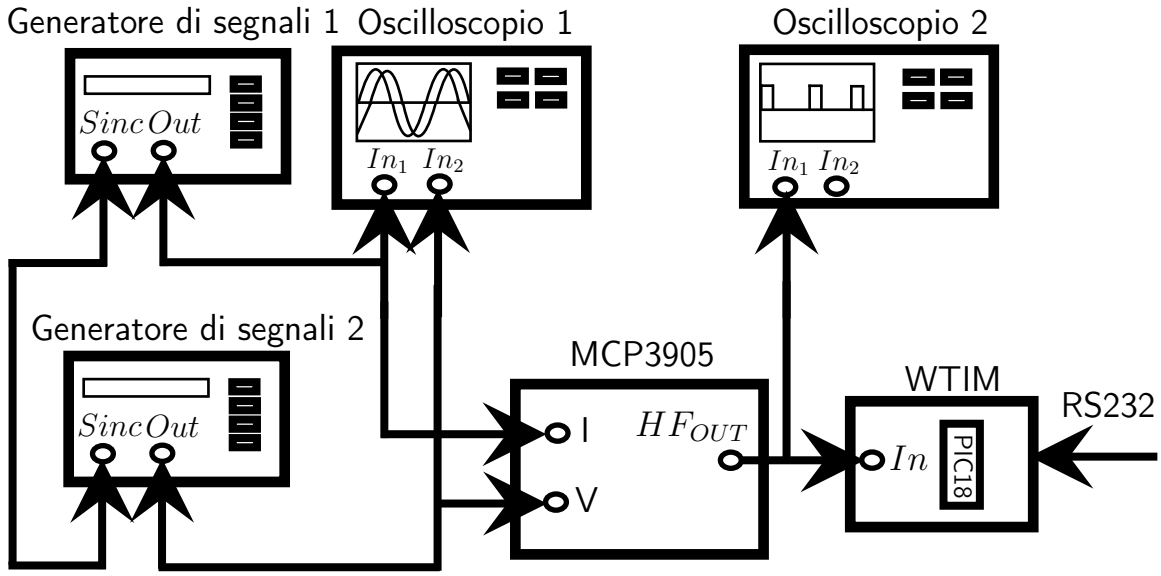

Figura 6.3: Schema di connessione con due generatori di segnale

6.2.3 Partitori resistivi agli ingressi per i generatori di segnale

Per poter connettere due generatori di segnale sincronizzati al fine di simulare gli esatti valori di tensione agli ingressi I e V si deve prima assicurare che, alla massima tensione possibile ($20\,V_{pp}$), non venga superata la soglia massima consentita.
Per l'integrato in questione all'ingresso V si ha un limite di $\pm 660\,mV$ e all'ingresso I vale $\pm 470\,mV$; si rende quindi indispensabile l'utilizzo e il dimensionamento di partitori di tensione aggiuntivi agli ingressi per garantire l'integrità dell'IC secondo il seguente schema:

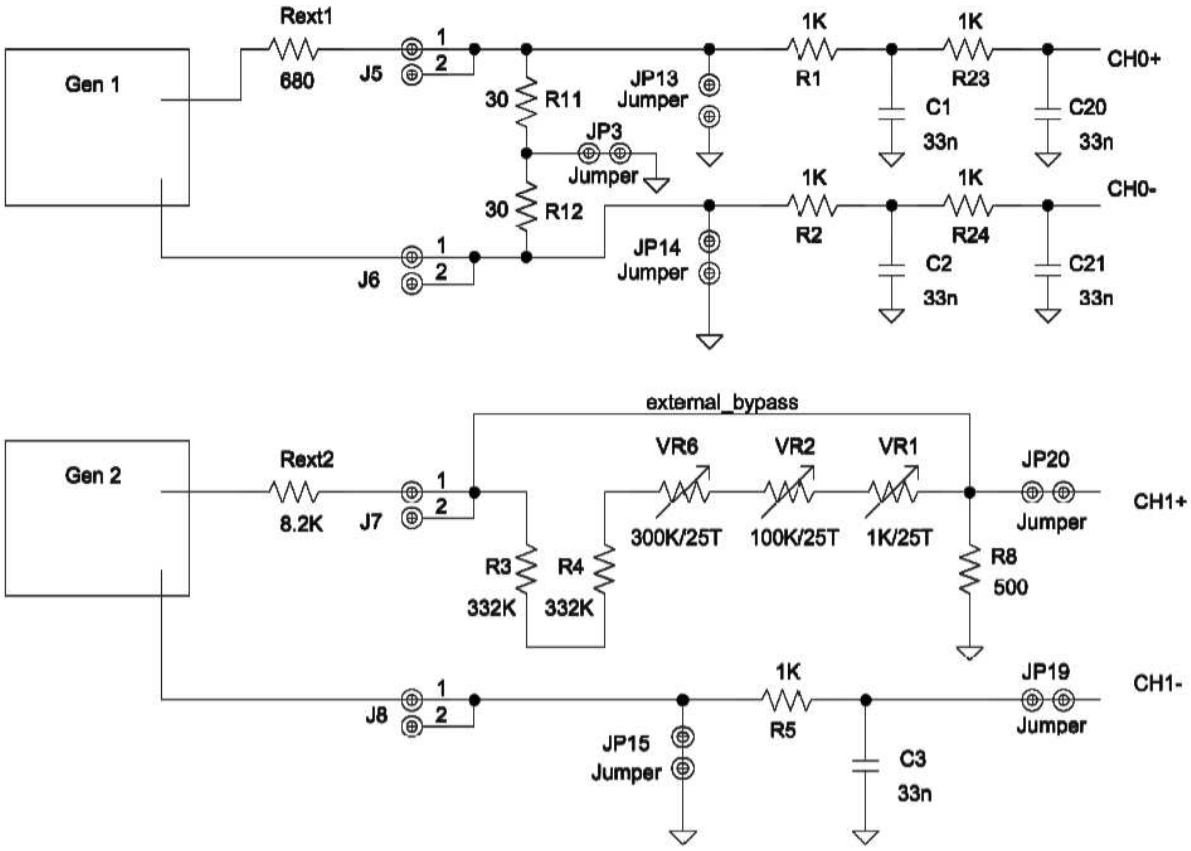

Figura 6.4: Schema di connessione partitori resistivi

In entrambi gli ingressi è già presente sulla scheda una resistenza tra ingresso e massa collegabile tramite jumper.
Agli ingressi I e V si trovano rispettivamente resistenze dichiarate di $30\,\Omega$ e $500\,\Omega$ e di valore effettivo misurato $30.2\,\Omega$ e $496.2\,\Omega$, a causa della tolleranza.
Per l'ingresso V si è bypassata la serie composta da resistenze e potenziometri di precisione necessari per l'utilizzo della scheda con la tensione di rete tramite un collegamento esterno.
Le resistenze da porre in serie agli ingressi I e V sono rispettivamente $680\,\Omega$ e $8200\,\Omega$, con valore effettivo misurato $680.8\,\Omega$ e $6758\,\Omega$.

Bisogna trasformare gli ingressi da differenziali a single-ended e di seguito sono elencati i jumper da riconfigurare:

Jumper	Valore	Note
JP3	Connesso	resistenza tra CH0 e massa
JP14	Connesso	CH0- a massa
JP15	Connesso	CH1- a massa

Tabella 6.3: Riconfigurazione dei jumper

I valori in tabella seguono le seguenti corrispondenze: $V_{I_{in}}$ è la tensione efficace all'ingresso CH0 dell'Energy Meter, $V_{I_{IC}}$ è la tensione efficace all'ingresso dell'IC, quindi dopo il partitore; $T_{teorico}$ è il periodo teorico calcolato secondo la formula 6.4, $T_{misurato}$ è il periodo letto dal WTIM, err e $err\%$ sono gli errori e $Potenza$ è il valore corrispondente di potenza di un ipotetico carico connesso all'Energy Meter.

n	$V_{I_{in}}$ $[mV_{rms}]$	$V_{I_{IC}}$ $[mV_{rms}]$	$T_{teorico}$ $[ms]$	$T_{misurato}$ $[ms]$	err $(T_{mis} - T_{teor})$	$err\%$ $(100 \cdot err/T_{teor})$	$Potenza$ $[W]$
1	3320	134.67	200.9	202	1.1	0.55	10951
2	2660	107.9	250.7	253	2.3	0.92	8775
3	2000	81.13	333.4	337	3.6	1.08	6599
4	1325	53.75	503.2	506	2.8	0.56	4372
5	660	26.91	1005.2	1013	7.8	0.78	2189
6	596	24.18	1118.6	1127	8.4	0.75	1967
7	466	18.9	1431.1	1444	12.9	0.9	1537
8	330	13.46	2009.6	2029	19.4	0.97	1095
9	199	8.11	3335.2	3385	49.8	1.49	660
10	68	2.76	9800.2	10059	214.8	2.19	224
11	33	1.37	19743.5	20259	515.5	2.61	111
12	19	0.85	31821.9	32868	1046.1	3.29	69

Tabella 6.4: Misure in funzione della potenza

Riporto in un grafico i valori di errore in percentuale per visualizzarne meglio l'andamento:

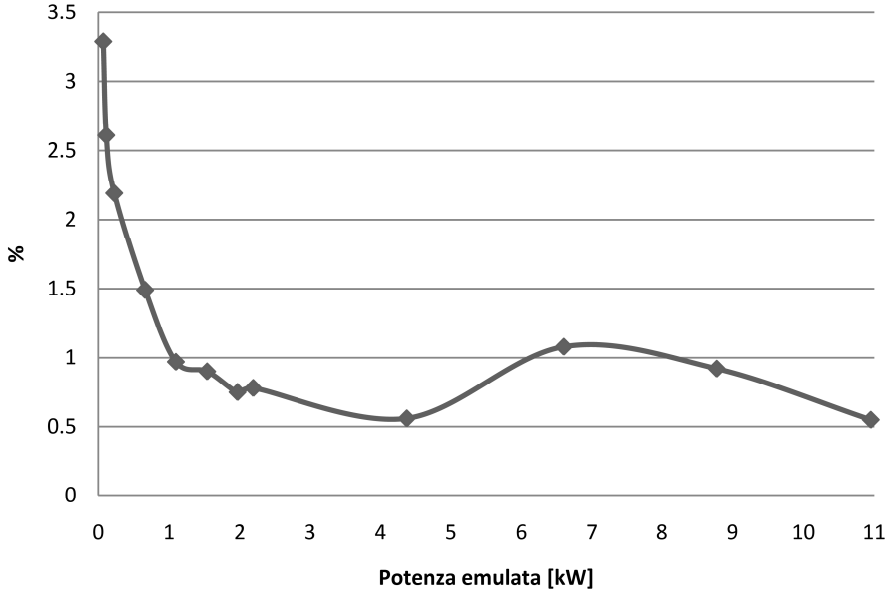

Figura 6.5: Accuratezza dell'emulazione di misura della potenza attiva

Da queste misure è evidente che lo scostamento tra l'indicazione del WTIM e il valore di riferimento non supera il $\pm 3.5\%$ per i periodi più grandi, cioè per potenze piccole. Questo risultato non sorprende, in quanto nel verificare l'accuratezza del WTIM come frequenzimetro si era giunti a un risultato analogo, infatti vediamo dalla colonna *err* di tabella 6.4 che la distanza tra la misura effettiva con l'oscilloscopio e il WTIM aumenta progressivamente come accade anche per la tabella 6.2.
Vi sono anche nuovi fattori che aumentano l'incertezza nella misurazione: il primo fattore riguarda lo sfasamento nullo che in realtà non è proprio uguale a zero: questo comporta una riduzione della frequenza di un fattore pari al coseno dello sfasamento. Il secondo fattore è l'incertezza della lettura negli oscilloscopi, perché si è preferito misurare il valore di tensione agli ingressi dell'Energy Meter piuttosto che direttamente all'ingresso dell'integrato MCP3905A[1].
Nella misura si è considerato anche l'attenuazione e lo sfasamento dovuti al doppio filtro RC in cascata all'ingresso di riferimento della corrente CH0, il filtro è previsto nella scheda per compensare l'effetto induttivo dello shunt che in questa configurazione non è presente. Per le ragioni appena citate lo sfasamento introdotto dal filtro RC contribuisce ad attenuare ulteriormente il segnale all'ingresso.

[1]Il motivo di questa scelta è pratico: infatti l'integrato MCP3905A è molto rumoroso nella parte analogica e ai suoi pin è presente un rumore con ampiezza molto elevata dovuto ai convertitori interni che impedisce una corretta misurazione della tensione.

6.2.4 Effetti dovuti allo sfasamento

Il grafico che illustra l'andamento della frequenza in funzione dello sfasamento fra tensione e corrente è riportato in fig. 6.6. In fig. 6.7 è riportato il grafico dello scostamento del coseno dello sfasamento misurato da quello teorico.

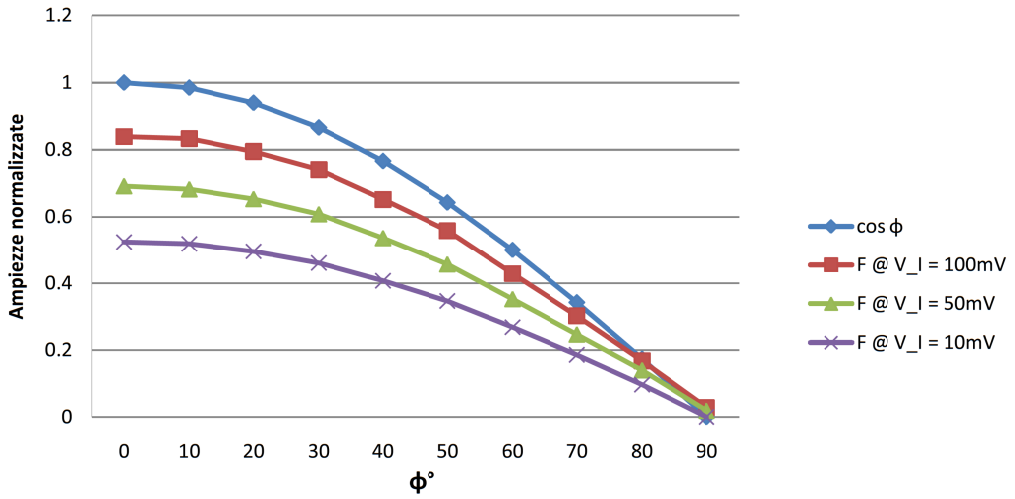

Figura 6.6: Accuratezza misura periodo in funzione dello sfasamento

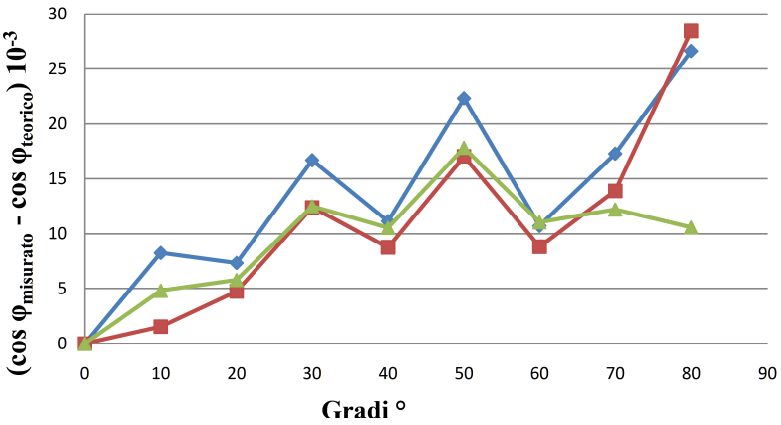

Figura 6.7: Scostamento dello sfasamento dal valore teorico

Questo grafico mostra le misure compiute dal WTIM utilizzando la configurazione di figura 6.3. In particolare è riportato l'andamento della funzione coseno al variare dello sfasamento che rappresenta il nostro riferimento e le tre misure su valori di tensione differenti. Questi ultimi sono stati normalizzati per distinguerne bene l'andamento indipendentemente dalla scala utilizzata. Con questa prova si vede anche dal grafico 6.6 che il valore di potenza misurata è direttamente proporzionale al coseno dello sfasamento. Anche a valori elevati di sfasamento si vede dal grafico 6.7 che lo scostamento è di un 3%

dal valore reale[1]. In Italia per esempio l'ENEL pone il vincolo di un $PF_{min} = \cos\varphi = 0.9$ per un'utenza domestica, il che equivale ad avere uno sfasamento massimo di $\varphi_{MAX} = \pm$ 25.84°.

φ	$cos(\varphi)$	T @ $V_I = 100\,\mathrm{mV_{rms}}$ [ms]	T @ $V_I = 50\,\mathrm{mV_{rms}}$ [ms]	T @ $V_I = 10\,\mathrm{mV_{rms}}$ [ms]
0	1	143	289	1526
10	0.98	144	293	1542
20	0.94	151	306	1614
30	0.86	162	329	1737
40	0.76	184	373	1965
50	0.64	215	438	2310
60	0.5	280	568	2986
70	0.34	398	812	4308
80	0.17	714	1430	8283
90	0	4311	10857	$>$ 60000

Tabella 6.5: Misure in funzione dello sfasamento

Di seguito si fornisce un ultimo quadro con quattro gruppi di misure in varie combinazioni potenza - sfasamento.
Le sigle V_I e V_V corrispondono alle tensioni impostate sui generatori di segnale e le sigle V_{CH0} e V_{CH1} alle tensioni all'ingresso CH0 e CH1 dell'Energy Meter.
Dalle tabelle 6.6, 6.7, 6.8 e 6.9 è evidente che le potenze misurate in questi quattro gruppi di misure differiscono in prima approssimazione di quasi un ordine di grandezza. L'errore rimane contenuto entro i limiti già riscontrati nelle precedenti verifiche e si presenta un aumento dell'errore all'aumentare dello sfasamento, sia esso in anticipo o in ritardo.

[1]Si ricorda che l'integrato MCP3905A fornisce in uscita un segnale con frequenza proporzionale alla sola potenza attiva, se il fattore di potenza $PF \neq 1$ vi è una riduzione della frequenza di un fattore $\cos\varphi$.

Primo gruppo: potenza corrispondente ≈ 43.3 kW

$V_I = 19.2\,\mathrm{V_{PP}}$; $V_V = 19.4\,\mathrm{V_{PP}}$; $V_{CH0} = 279.7\,\mathrm{mV_{rms}}$; $V_{CH1} = 459\,\mathrm{mV_{rms}}$

φ	$T_{teorico}$ [ms]	$T_{Oscilloscopio}$ [ms]	T_{WTIM} [ms]	Duty Cycle %	$err\%$ $100 \cdot (T_{WTIM} - T_{teorico})/T_{teorico}$
0	50.95	53.3	53	50	4.02
+54.72	88.21	99.4	99	50	12.22
−54.72	88.21	86.6	86	50	-2.51

Tabella 6.6: Misure di potenza attiva con sfasamento - 1

Secondo gruppo: potenza corrispondente ≈ 6.8 kW

$V_I = 7.75\,\mathrm{V_{PP}}$; $V_V = 7.69\,\mathrm{V_{PP}}$; $V_{CH0} = 113\,\mathrm{mV_{rms}}$; $V_{CH1} = 182\,\mathrm{mV_{rms}}$

φ	$T_{teorico}$ [ms]	$T_{Oscilloscopio}$ [ms]	T_{WTIM} [ms]	Duty Cycle %	$err\%$ $100 \cdot (T_{WTIM} - T_{teorico})/T_{teorico}$
0	324	336.5	336	26.7	3.7
+55.44	571.2	608	608	14.8	6.44
−55.44	571.2	580	579	15.4	1.37

Tabella 6.7: Misure di potenza attiva con sfasamento - 2

Terzo gruppo: potenza corrispondente ≈ 430 W

$V_I = 2\,\mathrm{V_{PP}}$; $V_V = 1.95\,\mathrm{V_{PP}}$; $V_{CH0} = 28.1\,\mathrm{mV_{rms}}$; $V_{CH1} = 44.9\,\mathrm{mV_{rms}}$

φ	$T_{teorico}$ [ms]	$T_{Oscilloscopio}$ [ms]	T_{WTIM} [ms]	Duty Cycle %	$err\%$ $100 \cdot (T_{WTIM} - T_{teorico})/T_{teorico}$
0	5176.6	5440	5420	1.8	4.7
+55.44	9125.5	9740	9709	1	6.4
−54.72	8962.7	9240	9206	1.1	2.7

Tabella 6.8: Misure di potenza attiva con sfasamento - 3

Quarto gruppo: potenza corrispondente ≈ 110 W

$V_I = 1.03\,\mathrm{V_{PP}}$; $V_V = 1.04\,\mathrm{V_{PP}}$; $V_{CH0} = 14.2\,\mathrm{mV_{rms}}$; $V_{CH1} = 23.2 mV_{rms}$

φ	$T_{teorico}$ [ms]	$T_{Oscilloscopio}$ [ms]	T_{WTIM} [ms]	Duty Cycle %	$err\%$ $100 \cdot (T_{WTIM} - T_{teorico})/T_{teorico}$
0	19944	21100	21052	<0.7	5.55
+54.72	34531	39500	39668	<0.8	14.88
−54.72	34531	30900	30910	<1	-10.49

Tabella 6.9: Misure di potenza attiva con sfasamento - 4

6.3 Risultati sperimentali

Dopo queste misure e tarature preliminari si collegano all'Energy Meter la tensione di rete, un wattmetro modello Siemens B4305 con accuratezza ±0.6% a 50 Hz e un carico, secondo lo schema di fig. 6.8.

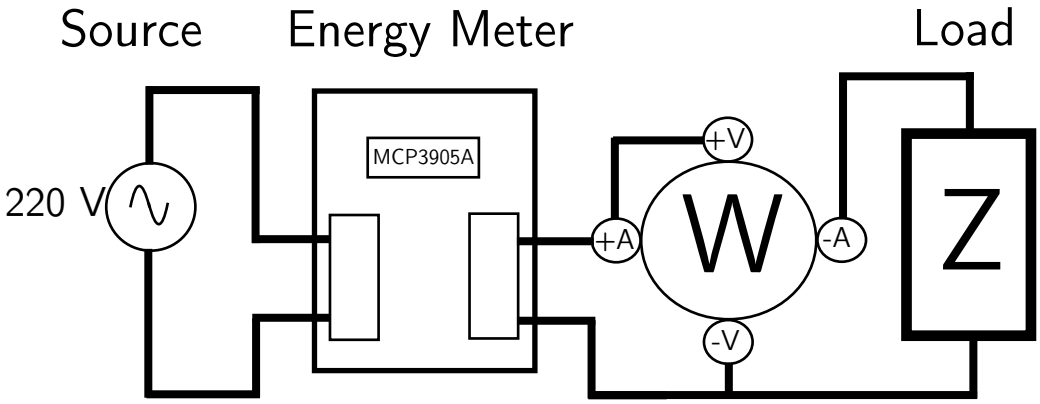

Figura 6.8: Schema di connessione con wattmetro

Nel grafico seguente si illustra l'andamento dell'errore in funzione della potenza attiva rilevato dalle misurazioni in tabella 6.10:

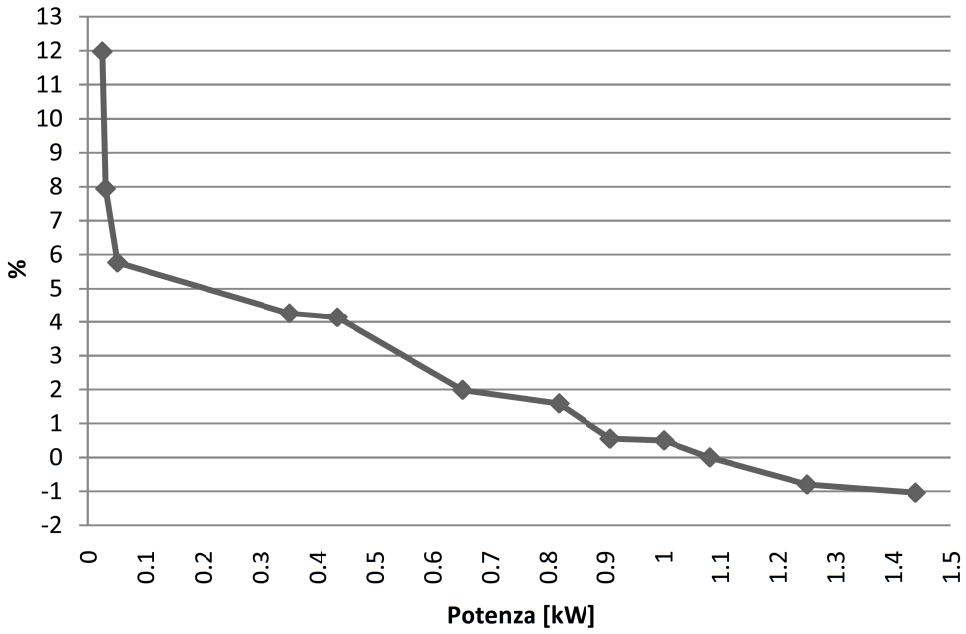

Figura 6.9: Grafico accuratezza misura finale

Il carico nel test è composto da più dispositivi e in particolare: una lampada alogena regolabile, due PC, un ventilatore e un riscaldatore per guaine termo-restringenti, quasi

tutti questi dispositivi dispongono di un regolatore di potenza interno che consente di avere a disposizione più livelli di potenza assorbita. Si riportano i risultati del test con questa configurazione e con carichi di diversa potenza in tabella 6.10.

n	Wattmetro [W]	Energy Meter [W]	err [W]	err% %
1	1440	1425	-15	-1.04
2	1252	1242	-10	-0.8
3	1083	1083	0	0
4	1002	1007	5	0.5
5	909	914	5	0.55
6	820	833	13	1.59
7	652	665	13	1.99
8	435	453	18	4.14
9	352	367	15	4.26
10	52	55	3	5.77
11	31.5	34	2.5	7.94
12	25.9	29	3.1	11.97

Tabella 6.10: Prove sperimentali

Dalla tabella 6.10 si può notare che la taratura dell'Energy Meter è stata eseguita per un valore di 1083 W in corrispondenza di un errore nullo nella lettura. Incrementando la potenza l'errore aumenta in modulo e il segno è negativo; al contrario diminuendo la potenza l'errore aumenta sempre in modulo, ma con segno positivo. Il motivo di questo comportamento appare nelle misure preliminari ottenute in fase di valutazione dell'accuratezza del WTIM come frequenzimetro: il WTIM ritarda progressivamente il numero di *ms* al diminuire della frequenza e quindi della potenza, come mostra la tabella 6.2. L'errore comunque rimane contenuto per valori di potenza medio - alti.

6.4 Note sul funzionamento del sistema

I problemi riscontrati in questo progetto sono fondamentalmente due; il primo riguarda il PIC18, nel quale si verifica la perdita della connessione se la lettura invocata dall'NCAP avviene in istanti vicini al fronte che abilita l'interrupt. Essendo l'interrupt di lettura a priorità più alta, probabilmente interferisce con le temporizzazioni dello stack ZigBee; questo inconveniente si è tuttavia verificato di rado nel corso delle misurazioni. L'altro problema è invece relativo all'NCAP e riguarda la manipolazione del dato ricevuto: in questo caso è inspiegabilmente accaduto che, dopo una serie di letture andate a buon fine, il microcontrollore e lo stack ZigBee si resettino.

Conclusioni

Questo progetto ha dimostrato come sia possibile realizzare uno *smart transducer* attraverso l'implementazione in un sensore di energia di un'interfaccia wireless basata su un microcontrollore PIC.
Lo standard IEEE 1451 permette la realizzazione di reti di sensori in modo da prescindere dal sistema fisico di comunicazione e i vantaggi che ne derivano sono molto significativi. In tale tesi si è scelto di implementare lo standard IEEE 1451 per realizzare una rete con dei sensori di energia; inoltre si è analizzato in maniera approfondita il sensore di energia, studiandone il comportamento al variare di diversi parametri, tra cui lo sfasamento fra i segnali di tensione e corrente.
Rimangono da risolvere due problematiche legate alla gestione degli interrupt che causano, seppure di rado, la perdita della connessione sia nel WTIM che nell'NCAP.
I TEDS, ovvero i Datasheets in formato elettronico, costituiscono l'altro punto di forza dello standard IEEE 1451: sono infatti le prime informazioni a essere lette dopo che un nuovo nodo viene connesso a una rete, liberando così la memoria dei nodi NCAP da grosse quantità di informazioni che riguardano i WTIM.
Si è dimostrato che la quantità di risorse richiesta in termini di memoria programma che risiede nella flash del microcontrollore lascia sufficiente spazio per inserire anche i TEDS al fine di rendere il WTIM autonomo, ossia con tutti i dati a bordo. Al momento i TEDS risiedono nell'NCAP, per riuscire a incrementare i diversi elementi software che compongono il sistema in modo indipendente.
Un possibile sviluppo futuro di tale progetto è la realizzazione di un sistema wireless per misurazioni della distribuzione della potenza in un impianto, disponendo in questo caso di molteplici Energy Meter. Tale soluzione sarà possibile nel momento in cui si supererà il limite attuale che non consente all'NCAP di creare una rete con più di un WTIM.
L'altro limite da superare consiste nella possibilità di connettere a una rete *Ethernet* l'NCAP, in modo che quest'ultimo possa interagire con il mondo esterno per esempio tramite *Internet*, per rendere disponibili le misurazioni su pagine *Web* tramite il *Web Server* interno.

Appendice A

Materiale utilizzato

Hardware

Di seguito descrivo brevemente tutto il materiale utilizzato.

Microchip EXPLORER 16 development board

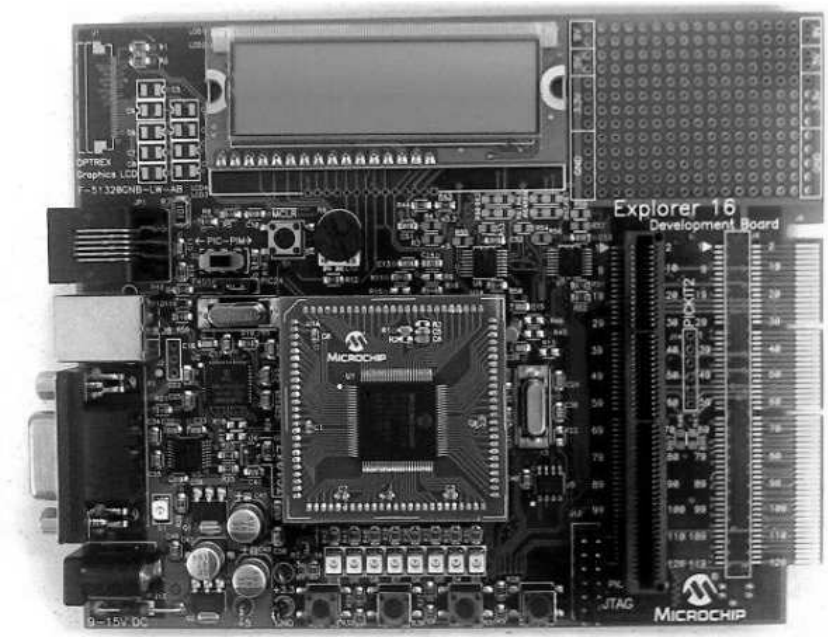

Figura A.1: Microchip EXPLORER 16

La scheda utilizzata nel progetto come CFD è composta dai seguenti elementi hardware:

- un microcontrollore PIC serie 24 (PIC 24FJ128GA010) su zoccolo da 100 pin;
- un quarzo a 8 MHz per il clock primario del microcontrollore;

- 4 pulsanti connessi alle porte I/O del microcontrollore;
- 8 LED connessi alle porte di I/O del microcontrollore;
- memoria EEPROM da 256 KByte con interfaccia seriale;
- sensore di temperatura;
- display LCD, 2 righe da 16 caratteri;
- potenziometro analogico;
- un connettore RJ-11 per la programmazione del firmware;
- un connettore RS-232 per la programmazione del firmare per l'output seriale;
- un connettore Jack femmina per l'alimentazione esterna.

Questa scheda di sviluppo ha a bordo un PIC 24FJ128GA010, di cui elenchiamo le caratteristiche:

- 128 KByte di memoria programma;
- 8 KByte di memoria RAM sincrona;
- 5 timer a 16 bit;
- 5 interfacce I/O di tipo capture / compare, con uscita PWM;
- 2 interfacce UART;
- 2 Serial Peripheral Interface (SPI);
- 2 interfacce I2C;
- un convertitore A/D a 10 bit, 16 canali;
- 2 comparatori.

Microchip PICDEM™Z evaluation board

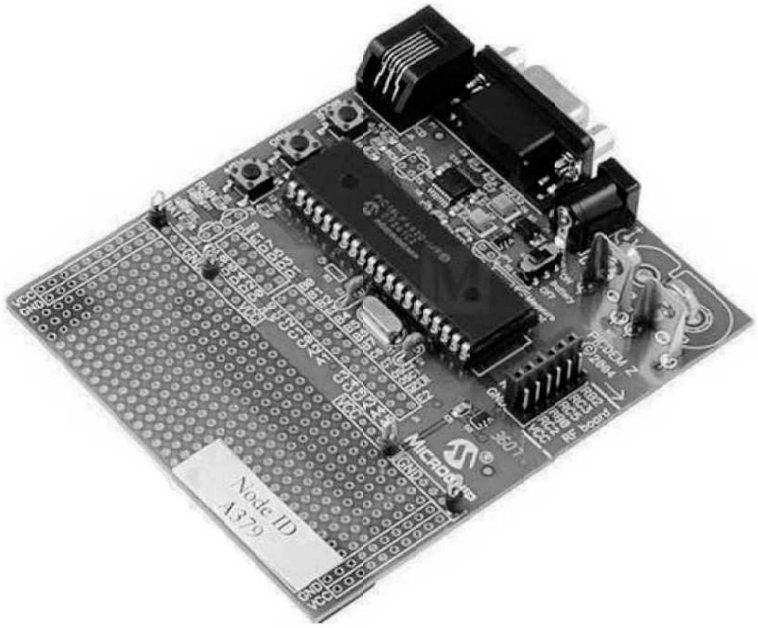

Figura A.2: Microchip PICDEM™Z

Il kit comprende una coppia di dispositivi uguali utilizzati come RFD, costituiti da una scheda madre che ospita:

- un microcontrollore PIC serie 18 (PIC 18LF4620) su zoccolo da 40 pin;
- un quarzo a 4 MHz per il PLL del microcontrollore che genera un clock a 16MHz;
- un sensore di temperatura Microchip TC77 a 5 pin connesso al microcontrollore tramite Serial Peripheral Interface (SPI), avente un range di funzionamento compreso tra -55 °C e +125 °C e con un errore massimo pari a 3 °C;
- 2 led collegati alle uscite digitali del microcontrollore;
- 2 pulsanti collegati agli ingressi digitali del microcontrollore;
- un pulsante di reset;
- un connettore RJ-11 per la programmazione del firmware;
- un connettore RS-232 per la programmazione del firmware e per l'output seriale;
- un connettore SPI per l'alloggiamento della scheda wireless;
- un connettore per l'alimentazione da batteria;

- un connettore Jack femmina per l'alimentazione esterna.

Il connettore SPI permette di utilizzare schede wireless diverse a seconda dell'uso che se ne vuole fare, mentre è possibile alimentare il tutto tramite una batteria a 9 V, in modo da rendere il dispositivo indipendente e trasportabile. Questa scheda di sviluppo ha a bordo un PIC 18LF4620 di cui elenchiamo le caratteristiche:

- PIC micro a 8 bit con frequenza di clock massima di 40 MHz;
- 64 KByte di memoria Flash per il firmware e per l'esecuzione dei programmi;
- 1024 KByte di memoria EEPROM;
- 3968 KByte di memoria RAM;
- 36 linee di I/O che possono essere utilizzate per la simulazione e la lettura di dati dall'esterno (sia in analogico che in digitale, grazie al convertitore A/D a 10 bit integrato).

Microchip MRF24J40MA PICDEM™Z 2.4 GHz RF BOARD

Le schede wireless con interfaccia SPI fornite di serie dal costruttore e montate sia sul PICDEM™Z che su EXPLORER 16 sono adatte alla banda dei 2.4 GHz e montano il trasmettitore/ricevitore (transceiver) Microchip MRF24J40 compatibile con lo standard IEEE 802.15.4. L'antenna è tracciata sul circuito stampato, ma c'è la possibilità di collegarne una esterna tramite un connettore SMA (da installare). Esistono altri transceiver compatibili, nati per essere utilizzati in bande di frequenza diverse e con standard diversi, ma in questa sperimentazione verrà utilizzato soltanto il transceiver sopra citato.

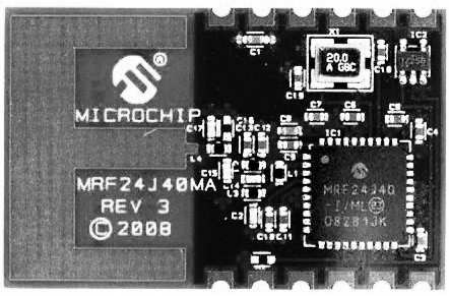

Figura A.3: Microchip MRF24J40MA RF BOARD

Microchip MPLAB ICD 2

Il Microchip MPLAB ICD 2 riunisce nello stesso dispositivo le caratteristiche di un In-Circuit Debugger (ICD) a basso costo e di un In-Circuit Serial Programmer (ICSP). Esso è studiato per funzionare da supporto nelle fasi di valutazione, debug e programmazione dei dispositivi in un ambiente di laboratorio. L'ICD 2 prevede diverse funzionalità, come l'esecuzione del codice in tempo reale e passo-passo, il monitoraggio e la modifica di variabili e registri, il debug direttamente nel circuito di prova, il monitoraggio della tensione di alimentazione del circuito, un led di diagnostica, un'interfaccia utente con Microchip MPLAB IDE e una connessione RS-232 o USB per il collegamento con il pc. Questo dispositivo permette di programmare i PIC Microchip attraverso l'interfaccia RJ-11 a sei poli senza dover togliere il microcontrollore dalla scheda. Questo comporta un notevole risparmio di tempo e una maggiore maneggevolezza del dispositivo, oltre al fatto che non dover rimuovere il microcontrollore previene eventuali danneggiamenti. Il Microchip MPLAB IDE consente di scegliere il dispositivo che si desidera utilizzare per il debug del circuito e per la programmazione. I menu Debugger e Programmer permettono di scegliere il device che si intende utilizzare per svolgere queste funzioni e, dopo una breve configurazione guidata, il dispositivo è già operativo e pronto all'uso. Il pulsante Reset and Connect to ICD consente al pc di ristabilire la comunicazione con ICD nel caso in cui si voglia ripristinare lo stato del dispositivo. Nella stessa barra degli strumenti i pulsanti dedicati al programmatore permettono di avviare la programmazione del dispositivo, che deve essere preventivamente collegato al Microchip MPLAB ICD 2 tramite il cavo RJ-11.

Figura A.4: Microchip MPLAB ICD 2

Quando il target device deve essere programmato o comunque gestito dal Microchip MPLAB ICD 2, esso è in uno stato di hold per cui, finché che non viene scollegato dal connettore a sei poli, il device target non si avvia normalmente. Senza dover scol-

legare il dispositivo è possibile utilizzare i pulsanti Relase from Reset e Hold in Reset presenti nella barra dedicata al programmatore di MPLAB IDE, per passare dallo stato di esecuzione normale allo stato di gestione da parte del Microchip MPLAB ICD 2. Una finestra di dialogo mostra l'avanzamento della programmazione e riporta il risultato della stessa, indicando se essa è andata a buon fine oppure gli eventuali errori riscontrati.

Microchip Energy Meter

La scheda sottostante realizza la misura di potenza ed energia:

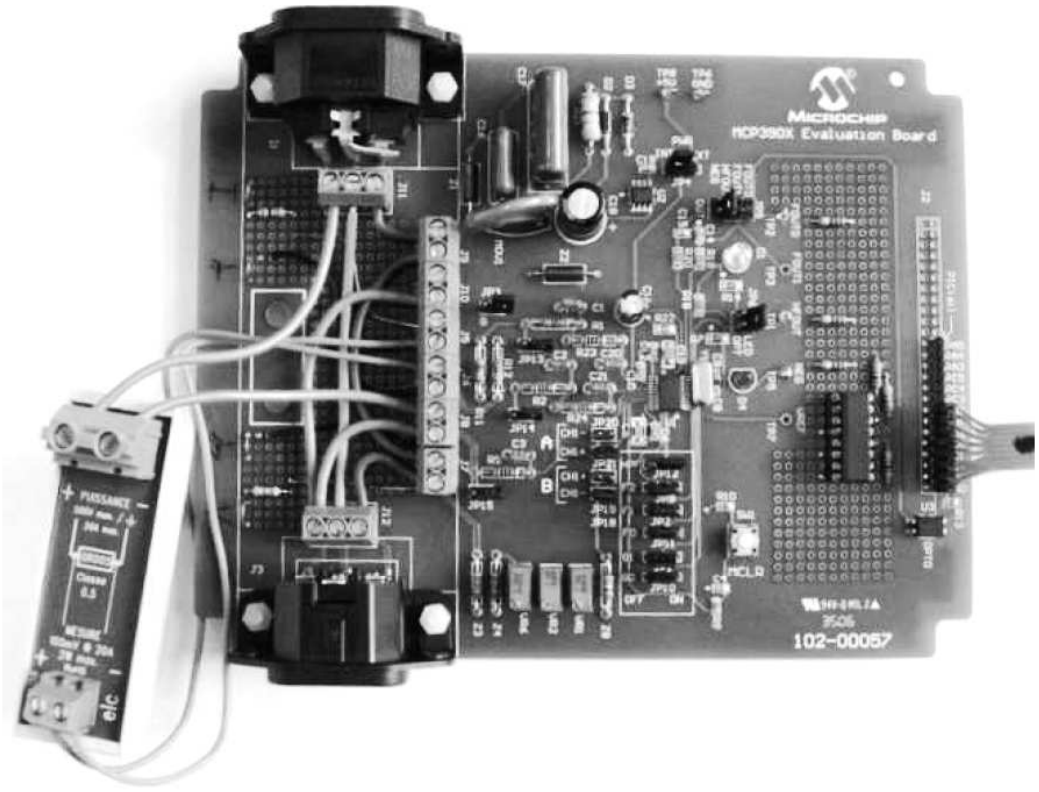

Figura A.5: Microchip Energy Meter

Essa è dotata di due connettori a cui vanno collegati la tensione di rete e il carico. Dalla lunga morsettiera verticale si preleva la tensione di rete per l'alimentazione della scheda, così come la tensione da misurare e da inviare in ingresso all'integrato. Si noti dalla figura A.5 il particolare dello shunt esterno connesso alla scheda Energy Meter. I tre potenziometri servono per la taratura della scheda: a partire dal trimmer più a sinistra la regolazione diventa sempre più precisa, perciò per una prima regolazione della scheda si interverrà sul primo trimmer a sinistra. Il led rosso indica la presenza di uno sfasamento negativo tra i due canali superiore a 90°. Nella parte di sviluppo è stato aggiunto un op-

toisolatore quadruplo modello PC845XJ0000F Sharp che consente di trasferire all'esterno tutti i segnali di uscita dell'MCP3905A, garantendone l'isolamento galvanico. Infine vi sono i jumper, che permettono la taratura e il settaggio delle diverse funzionalità della scheda.

Software

Microchip MPLAB IDE

Microchip MPLAB IDE è una suite per PC che serve a sviluppare applicazioni per microcontrollori Microchip. Questo programma è un Integrated Development Environment (IDE), in quanto fornisce un singolo ambiente integrato per sviluppare un codice per microcontrollori embedded. Ciò che caratterizza i sistemi embedded è il fatto che essi combinano le potenzialità di un semplice microprocessore con alcuni circuiti, chiamati periferici, integrati nello stesso chip.

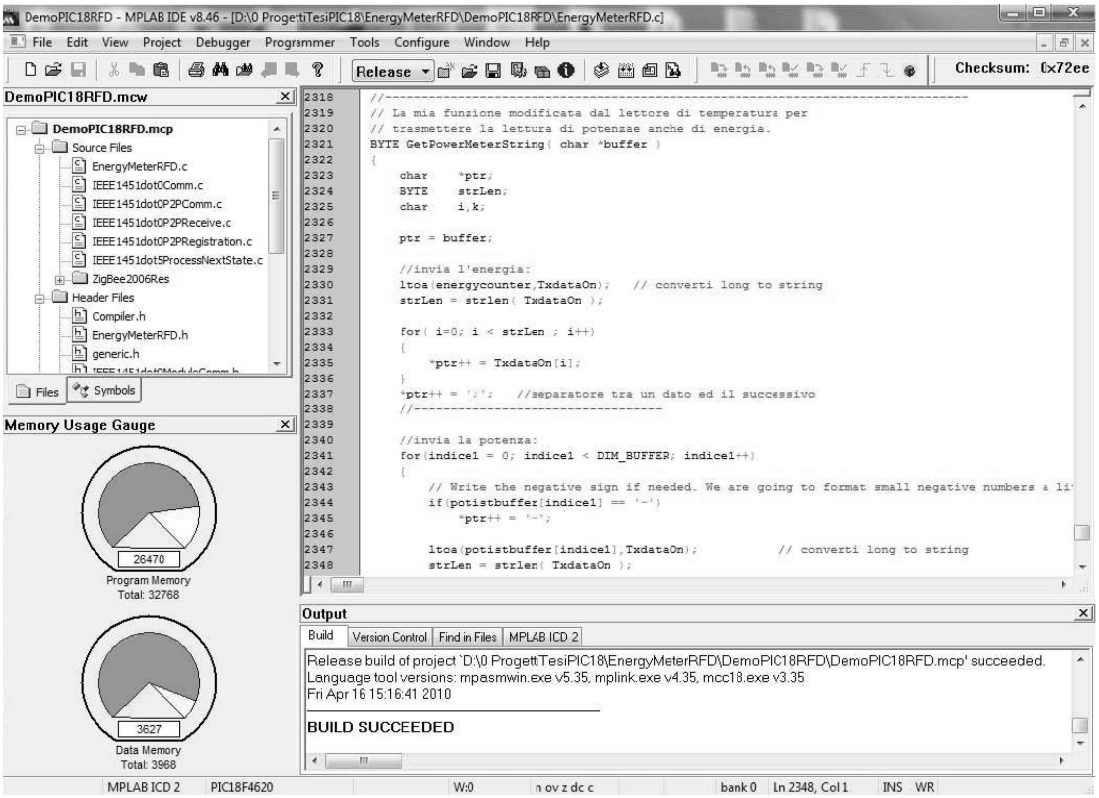

Figura A.6: Microchip MPLAB IDE

Questo porta ad avere un microcontrollore che necessita di pochi componenti esterni aggiuntivi per funzionare. Scrivere un'applicazione per questi sistemi richiede un idoneo ambiente di sviluppo: MPLAB IDE è l'ambiente fornito da Microchip. Esso permette di

seguire tutte le fasi di sviluppo dell'applicazione e contiene un editor per la stesura del codice, un project manager per gestire i file sorgente e le impostazioni, un compilatore per convertire il codice sorgente in linguaggio macchina e un dispositivo per la simulazione e la gestione di un programmatore hardware che trasferisce il codice macchina al microcontrollore.

Microchip MPLAB C-18 Compiler

L'MPLAB C-18 Compiler è un add-on per l'ambiente di sviluppo MPLAB sopra descritto che consente la compilazione di programmi scritti in linguaggio ANSI C per microprocessori a 8 bit della serie Microchip PIC18.

HI-TECH PIC24/dsPIC C Compiler

L'MPLAB C-30 Compiler è un add-on per l'ambiente di sviluppo MPLAB sopra descritto e permette la compilazione di programmi scritti in linguaggio ANSI C per microprocessori della serie Microchip PIC24.

Microsoft Hyper Terminal

Si tratta di un'applicazione per la gestione e la visualizzazione delle comunicazioni seriali, utilizzata per visualizzare i messaggi su linea seriale RS232 generati dai dispositivi NCAP e TIM. Le impostazioni impiegate in questo progetto per tutti i dispositivi sono:
velocità = 19200 baud, bit dati = 8, controllo parità = N (nessuno) e bit stop = 1.

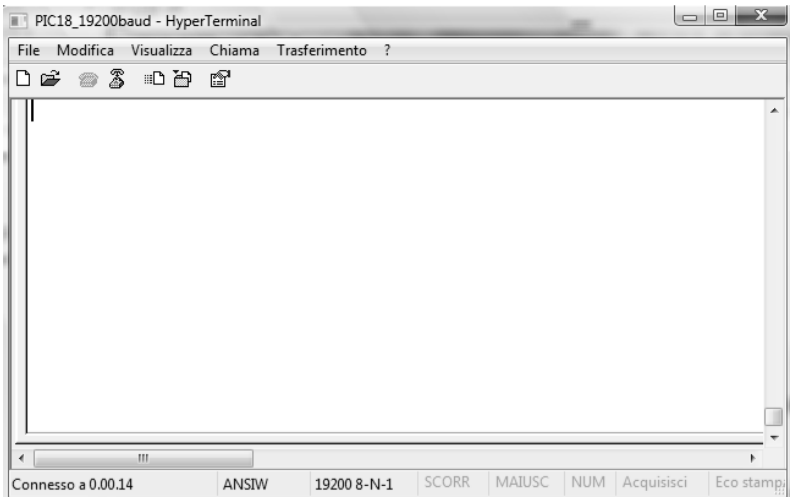

Figura A.7: Microsoft Hyper Terminal

Bibliografia

1. *AN965-Microchip Stack for the ZigBee Protocol*, Microchip Technology Inc., 2006.
2. Barbon N., *Studio di implementazione del protocollo IEEE 1451 su reti di sensori ZigBee*, Università di Padova, 2008.
3. Bergamin F., *Procedure di interfacciamento di sensori wireless secondo gli standard IEEE 1451.5 e ZigBee*, Università di Padova, 2008.
4. *Explorer 16 Development Board User's Guide*, Microchip Technology Inc., 2005.
5. Giorgi G., Microsoft PowerPoint (slide1451.ppt), Università di Padova, 2008.
6. Guarnieri M., Stella A., *Principi ed applicazioni di elettrotecnica*, ed. Progetto, Università di Padova, 2001.
7. *MCP3905/06 Energy-Metering ICs with Active (Real) Power Pulse Output Data Sheet*, Microchip Technology Inc., 2009.
8. *MCP3905A/06A Evaluation Board User's Guide*, Microchip Technology Inc., 2009.
9. *PC845XJ0000F Series-DIP 16 pin (4-channel) Darlington Phototransistor Output, Photocoupler*, SHARP Corporation, 2006.
10. *PIC18F2525/2620/4525/4620 Data Sheet, 28/40/44-Pin Enhanced Flash Microcontrollers with 10-Bit A/D and nanoWatt Technology*, Microchip Technology Inc., 2007.
11. *PIC24FJ128GA010 Family Data Sheet, 64/80/100-Pin General Purpose, 16-Bit Flash Microcontrollers*, Microchip Technology Inc., 2007.
12. *PICDEM™Z Demonstration Kit User's Guide*, Microchip Technology Inc., 2008.
13. Song E.Y., Lee K., *Understanding IEEE 1451 Networked Smart Transducer Interface Standard*, 2008.

Ringraziamenti

Grazie ai miei genitori che con il sostegno dimostrato mi hanno permesso di raggiungere questo importante traguardo.

Grazie a Elena per i bei momenti passati insieme, per il sostegno in tutti i momenti difficili tra università e musica e per l'aiuto nel correggere questa tesi.

Grazie agli amici Stefano Negro e Marco Serpelloni per aver reso più piacevole il tempo passato all'università e non solo.

Grazie a tutti gli amici nel mondo della musica.

Grazie ai colleghi del laboratorio, ing. Marco Stellini, ing. Alberto Bellato, ing. Pierpaolo Russo e a Filippo Santello, per aver reso il periodo in laboratorio più vivace e divertente.

Grazie al prof. Claudio Narduzzi per avermi permesso di approfondire la programmazione dei PIC nonché la realizzazione di questo interessante progetto.

Grazie alla musica per avermi insegnato che la tenacia e l'impegno sono i due fattori essenziali e determinanti per raggiungere tutti gli obbiettivi!

{♯ ♮ 𝅘𝅥𝅯 𝄫 ♮ ♭ }