UNIVERSITÀ DI PADOVA **SCUOLA DI INGEGNERIA**

TESI DI LAUREA

MISURE DI PRESTAZIONI DI SISTEMI DI COMUNICAZIONE POWERLINE NELLA BANDA CENELEC C

Laureando: *Alessio Fusaro*

Relatore: Stefano Vitturi
Correlatore: Federico Tramarin

Corso di Laurea Magistrale in Ingegneria dell'Automazione

13 Ottobre 2015

Anno Accademico 2014/2015

Prefazione

La trasmissione dei dati attraverso le linee della fornitura elettrica basata sul principio delle onde convogliate trova origine circa negli anni 20. Venne utilizzata per la prima volta in Europa per la trasmissione di dati di controllo a lunga distanza attraverso le linee di alta tensione nelle frequenze 15 kHz - 500 kHz. Tutt'ora le Power-Line Communications (PLC) vengono utilizzate in determinati range di frequenze per vari scopi, dal controllo a distanza dei carichi ad esempio nell'ambito dell'illuminazione pubblica alla telelettura dei contatori, dai sistemi di automazione domestica fino alle connessioni a banda larga per reti LAN e la fornitura del servizio internet. Il pregio principale di questa tecnologia consiste nella possibilità di sfruttare i collegamenti delle linee elettriche già esistenti per effettuare trasmissione di dati, eliminando le spese relative ad ulteriori cablaggi altrimenti necessari. Lo sviluppo tecnologico degli ultimi decenni inoltre permette, per le applicazioni domestiche a bassa velocità, la realizzazione di sistemi di questo tipo a prezzi molto contenuti.

Il lavoro svolto e descritto in questa tesi consiste nell'analisi delle prestazioni delle PLC in banda stretta per applicazioni domotiche, individuata e regolata in Europa dallo standard CENELEC EN5006; in particolare si è studiato il funzionamento di un chip che lavora nella Banda C che va dai 125 kHz ai 140 kHz.

Nel primo capitolo vengono introdotte le comunicazioni powerline e mostrata una panoramica generale della distinzione delle frequenze.
Nel secondo capitolo sono illustrate le caratteristiche principali e le potenzialità del chip utilizzato in questo lavoro di tesi, il Cypress CY8CPLC20.
Nel terzo capitolo vengono principalmente presentati i software realizzati ed utilizzati per l'analisi delle prestazioni, argomento centrale della tesi.
Nel quarto capitolo sono mostrati e discussi i risultati ottenuti sperimentalmente con la rete priva di disturbi, analizzati in particolare dal punto di vista della tempistica.
Nel quinto invece vengono analizzate le prestazioni in termini di $packet\ error\ rate$ e percentuali di successo nei test di polling quando nella rete viene inserito un disturbo causato da un inverter connesso a dei pannelli solari.
Nel sesto capitolo infine sono presentate le conclusioni relative al lavoro svolto.

La tesi è stata realizzata nell'ambito di una collaborazione tra CNR-IEIIT e la ditta KBlue.

Ringraziamenti

Desidero ringraziare:

l'azienda KBlue,
in particolare l'ing. Roberto Tisato,
per l'opportunità data.

F. Tramarin, M. Stellini e T. Caldognetto,
per l'aiuto e la disponibilità dedicatami in questi mesi.

Luca,
per lo spirito con cui abbiamo condiviso
questi ultimi anni di Università.

Il professor Vitturi,
per la simpatia ed il clima piacevole
con il quale mi ha permesso di lavorare.

Gli Anni Azzurri, i Baracchini e gli Sbueeels,
per il divertimento garantito in tutte le occasioni
ed i traguardi indimenticabili di fine estate.

Giulia,
per aver sopportato anche la versione fantasma di me,
capendomi come una sorella.

Alice,
per aver reso i miei problemi universitari più piccoli
e questi mesi assieme a te tra i più belli della mia vita.

Annamaria e Giancarlo,
per avermi sempre sostenuto
e dimostrato il vostro orgoglio.

Nicolas,
per esserci sempre
e per aver dato tregua alla guerra in casa tra lavoratori e studenti
riscoprendo una nuova amicizia.

Roberta e Nicolò,
per la pazienza con cui mi avete capito ed accettato,
dedico a voi la mia laurea, il mio passato, il mio presente ed il mio futuro.

Indice

Elenco delle figure

Elenco delle tabelle

Capitolo 1

Introduzione

1.1 Comunicazioni Powerline

Le comunicazioni powerline, abbreviate in PLC (dall'inglese, Power-Line Communications), consistono nella trasmissione di dati codificati attraverso le linee elettriche mediante l'uso delle onde convogliate. Ciò consiste nel sommare alla tensione AC 230 V a 50 Hz un segnale di ampiezza molto inferiore e frequenza superiore modulata dai dati, come mostrato in figura 1.1.

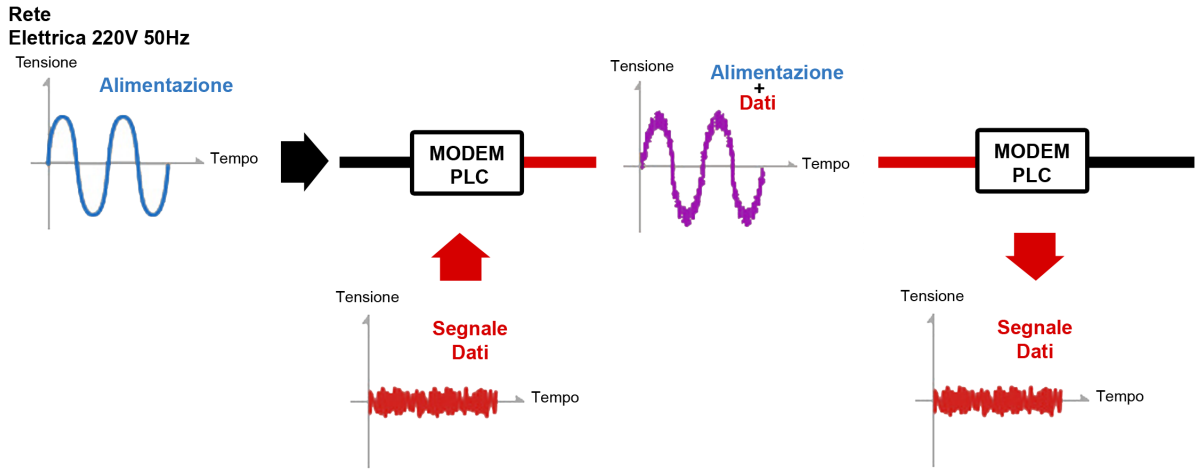


Figura 1.1: *Principio di funzionamento delle onde convogliate*

Con questa tecnologia è possibile trasferire dati attraverso i cavi della fornitura elettrica e raggiungere distanze e velocità di trasmissione molto interessanti. La comodità del poter sfruttare i collegamenti della rete elettrica già esistente, senza doverne aggiungere di nuovi, evidenzia le potenzialità delle PLC sia in ambito domestico che non, comportando facilmente importanti riduzioni dei costi. Le PLC possono essere utilizzate per vari scopi, ad esempio per la fornitura del servizio internet presso luoghi isolati (non raggiunti dalla rete telefonica ma solamente da

quella elettrica), per implementare tecnologie di automazione domestica o per controlli remoti di qualsiasi tipo, in tutti i casi evitando l'installazione di cablaggi aggiuntivi. Nonostante l'utilizzo di questa tecnologia non abbia ancora sostituito altri metodi di trasmissione più classici, non si tratta di un'invenzione recente. Già nel 1922 in Europa venne sviluppato il primo sistema di trasmissione mediante onde convogliate con frequenze dai 15 kHz ai 500 kHz. Tutt'ora questo tipo di comunicazione viene utilizzato da alcune compagnie elettriche per il controllo ad esempio dell'illuminazione pubblica o di sistemi di gestione dell'energia. Ad esempio sono diffusi soprattutto in America i cosìdetti "Baby Phones" che trasmettono un segnale analogico di bassa qualità per la voce attraverso i normali collegamenti a 230 V.
Le potenzialità di una trasmissione digitale PLC in ambito domestico invece risiedono ad esempio nella possibilità di far comunicare fra loro gli elettrodomestici per una miglior gestione dell'energia o di poter controllare l'accensione e la regolazione di luci, tapparelle, riscaldamento, condizionamento e qualsiasi altro dispositivo connesso alla rete elettrica. Le PLC indoor e quelle outdoor tuttavia non sono uguali e differiscono principalmente per la disponibilità delle bande di frequenza e per i livelli massimi del segnale consentiti.
Per quanto riguarda le trasmissioni PLC si distinguono le seguenti comunicazioni, illustrate anche in Figura 1.2:

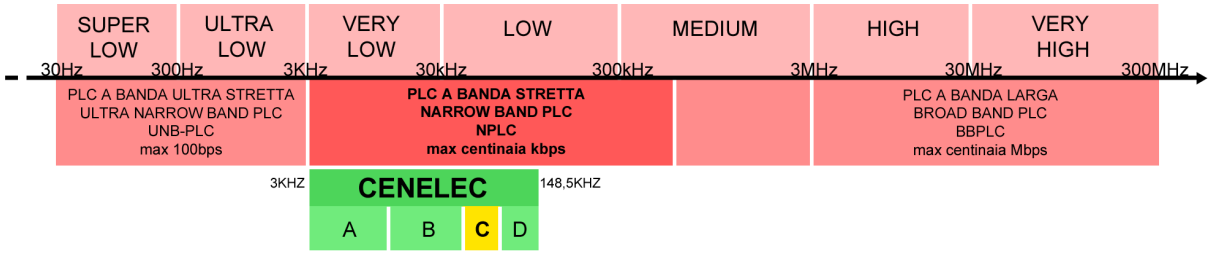


Figura 1.2: *Tipologie di PLC nelle varie frequenze*

- **PLC a Banda Ultra Stretta**, Ultra Narrow Band PLC, UNB-PLC
  Ultra Low Frequency, 300-3000 Hz, e parte superiore delle Super Low Frequency, 30-300 Hz
  basse velocità di comunicazione, massimo 100bps
  comunicazioni a lunga distanza in grado di attraversare trasformatori e evitare ripetitori
  utilizzate in America in particolare per sistemi di lettura remote e controllo diretto del carico mediante comunicazioni TWACS, Two-Way Automatic Communication System

- **PLC a Banda Larga**, Broadband PLC, BB-PLC
  High Frequency / Very High Frequency, 1.8-250 MHz
  alte velocità di comunicazione, dalle decine di bps fino alle centinaia di Mbps
  è stato provato l'utilizzo in America anche per la fornitura del servizio internet ma il progetto non è andato a buon fine e si è concluso nel 2008 a causa delle difficoltà nella trasmissione dei dati e dei disturbi causati alle comunicazioni dei radioamatori

- **PLC a Banda Stretta**, Narrowband PLC, N-PLC,
  Very Low Frequency/Low Frequency/Medium Frequency, 3-500 kHz
  comunicazioni dell'ordine massimo delle centinaia di Kbps
  che comprendono:
  - frequenze CENELEC (Comitato Europeo di Normalizzazione Electrotecnica) per l'Europa (3-148.5 kHz)
  - frequenze FCC (Federal Communications Commission) per l'America, 10-490 kHz
  - frequenze ARIB (association of Radio Industries and Businnesses) per il Giappone, 10-450 kHz
  - frequenze cinesi 3-500 kHz

Per quanto riguarda la banda larga, allo stato dell'arte sono reperibili in commercio a prezzi inferiori di 70 euro, dispositivi utilizzabili per i collegamenti di reti LAN ed internet che sfruttano il range di frequenze tra i 2 e i 50 MHz, in grado di garantire trasmissioni fino a 200 Mbps e distanza massima 300 m (la velocità effettiva tuttavia può diminuire anche di molto a causa della difficile modellizzazione del mezzo che dipende moltissimo dalla struttura della rete utilizzata, dai materiali, dai collegamenti, dal rumore e da altri fattori tecnici.)

Le comunicazioni indoor a bassa frequenza invece avvengono con una velocità massima di 2400 bps e utilizzano tecnologie a basso costo. Questa velocità, seppur non molto alta, risulta soddisfacente per l'utilizzo in applicazioni domotiche dove la quantità di dati trasmessi e le velocità di trasmissione richieste spesso non sono elevate poiché nella maggior parte dei casi si tratta di segnali di controllo del tipo ON/OFF o regolazione di cursori. In Figura 1.3 si può osservare un esempio di sistema di automazione domestica che sfrutta comunicazioni PLC.

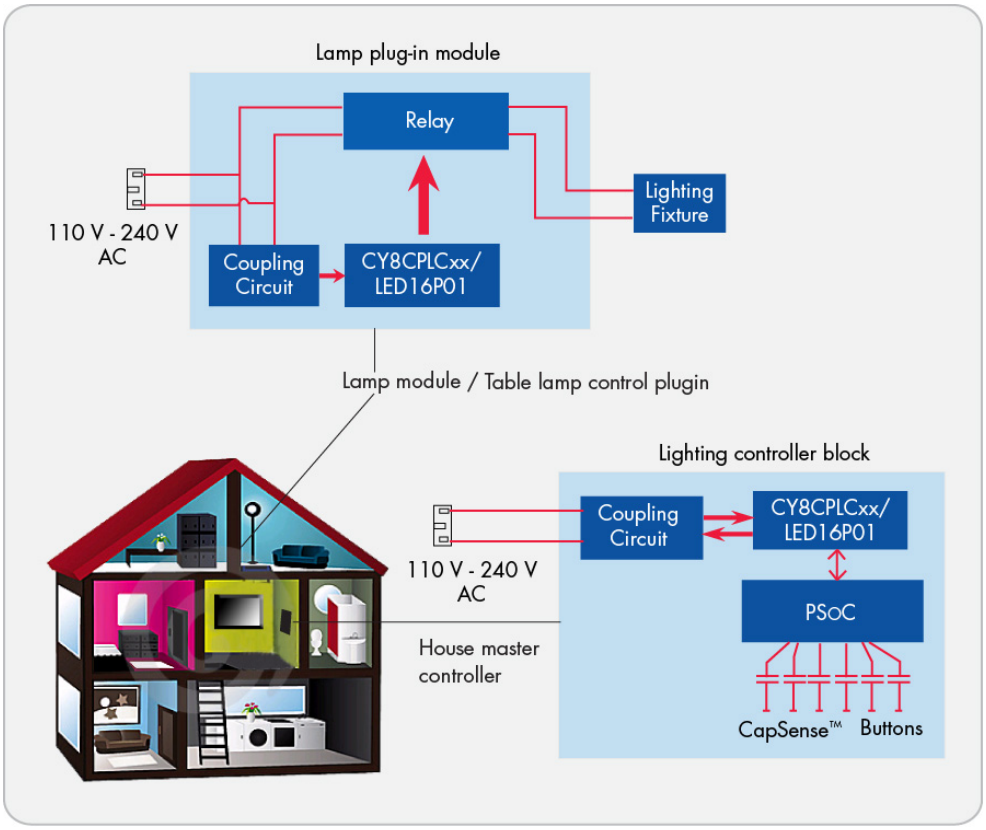


Figura 1.3: *Esempio di sistema domotico PLC*

Per quanto riguarda le normative europee, il Comitato Europeo di Normazione Elettrotecnica (CENELEC) ha stabilito, come accennato in precedenza, lo Standard $EN\ 50065-1$ "*Signalling on low-voltage electrical installations in the frequency range 3 kHz to 148.5 kHz*" relativo alla trasmissione di segnali in installazioni elettriche a basso voltaggio nel range di frequenze che va appunto dai 3kHz ai 148,5 kHz. In particolare vengono distinte le quattro bande di frequenze evidenziate in Figura 1.4 per le quali sono definiti i seguenti relativi scopi di utilizzo e metodi di accesso al mezzo:

- **Banda A**, da 3 kHz a 95 kHz, assegnata alle compagnie di distribuzione di energia elettrica
- **Banda B**, da 95 kHz a 125 kHz, ad uso comune, nessun protocollo di accesso
- **Banda C**, da 125 a 140 kHz, ad uso domestico con specifico protocollo di accesso CSMA/CA (Carrier Sense Multiple Access/Collision Avoidance) per facilitare la coesistenza di più sistemi nella stessa rete e che lavorano nella stessa banda di frequenze
- **Banda D**, da 140 kHz a 148,5 kHz, per sistemi di sicurezza e allarme, nessun protocollo di accesso

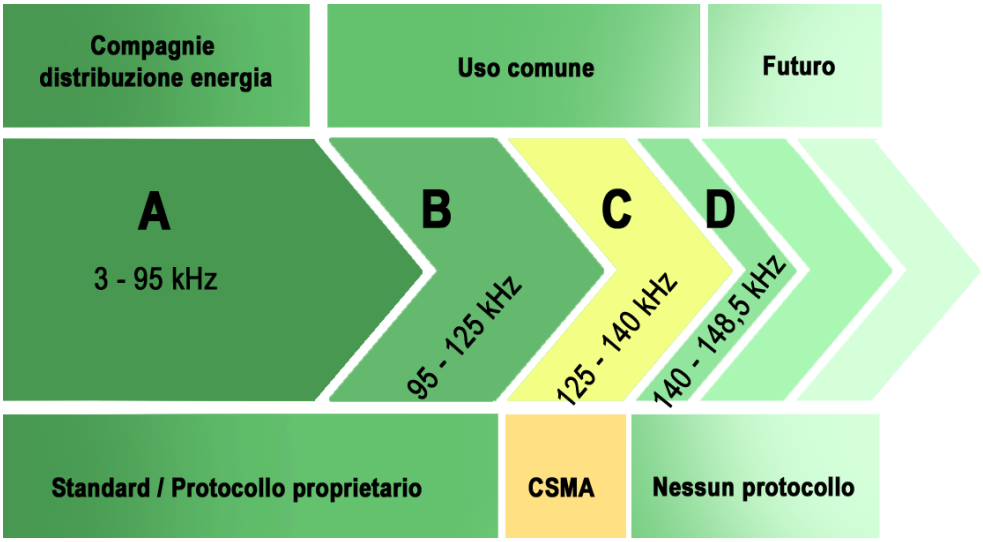


Figura 1.4: *Suddivisione delle frequenze nella Banda CENELEC*

Per quanto riguarda i limiti di potenza di ogni banda invece si fa riferimento alla Figura 1.5

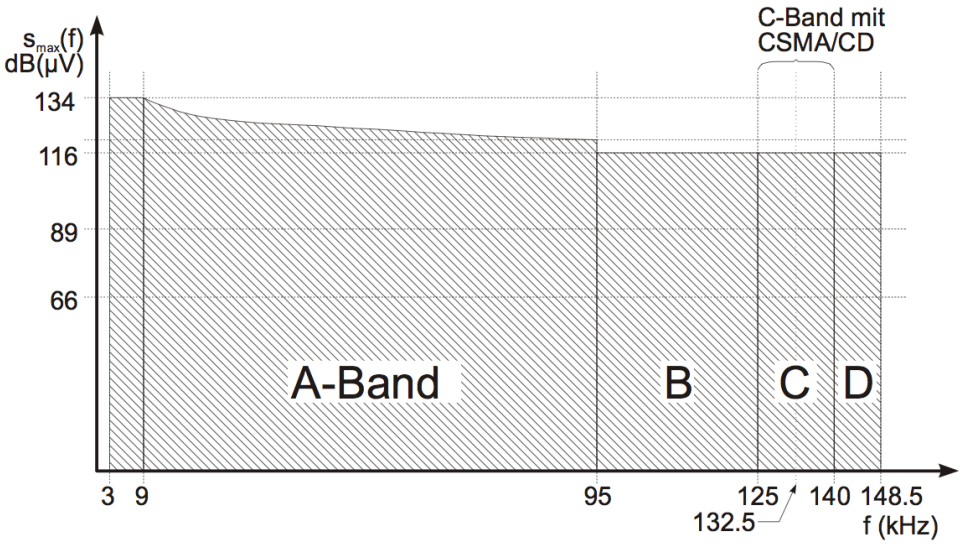


Figura 1.5: *Limiti di potenza nelle bande CENELEC*

Nei sistemi in cui il metodo di accesso è del tipo CSMA/CD (Carrier Sense Multiple Access/Collision Detection), ogni dispositivo deve controllare la banda di frequenze per un certo intervallo prima di inviare un pacchetto. Questo serve per evitare possibili collisioni nel caso in cui altri dispositivi nella rete stessero già utilizzando la stessa banda di frequenze. CENELEC per la Banda C stabilisce anche i limiti per:

- il tempo massimo di occupazione di un dispositivo con relative pause massime: max 1 secondo di trasmissione con max 80 ms di pausa tra un pacchetto e l'altro
- il tempo minimo di pausa tra due invii dello stesso dispositivo: min 125 ms
- il tempo tra la fine di un invio di un dispositivo e l'inizio di quello di uno di un altro: min 85 ms

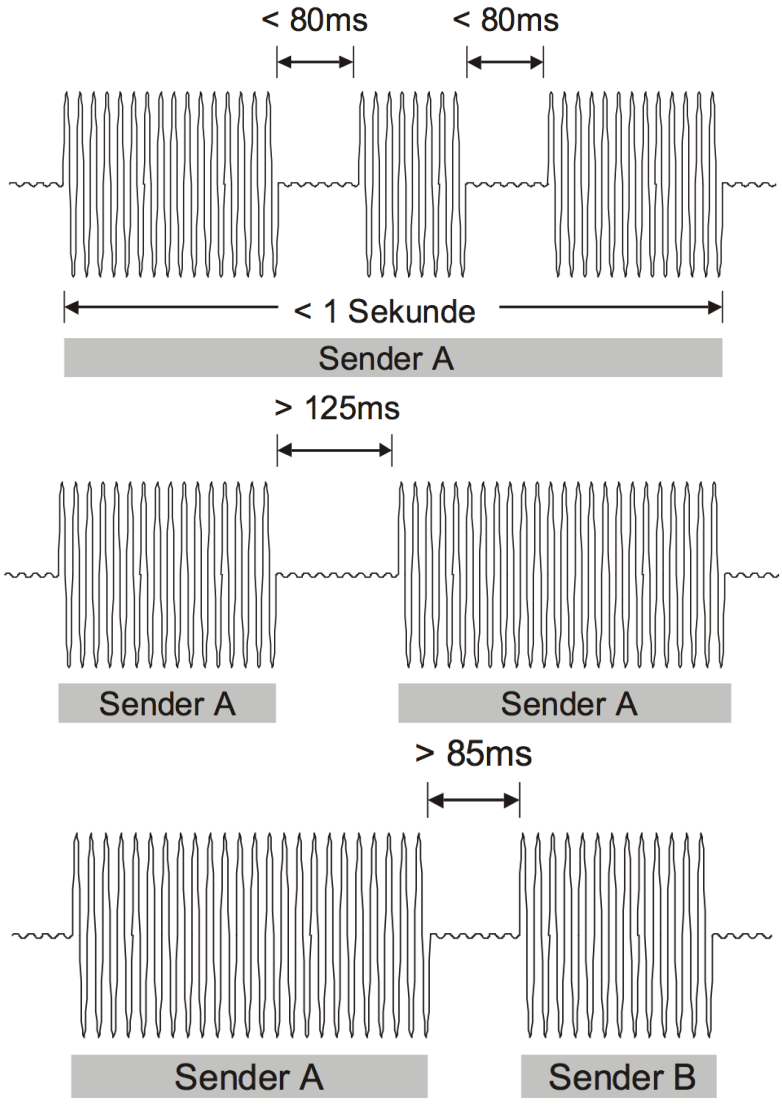


Figura 1.6: *Limiti relativi ai tempi nella banda CENELEC-C*

Il lavoro di tesi svolto consiste nell'analisi delle prestazioni di un chip PLC in particolare, descritto nel prossimo capitolo, che lavora nella banda CENELEC-C.

Capitolo 2

Il Modem Cypress CY8PLC20

2.1 Introduzione

Per l'analisi delle prestazioni delle comunicazioni powerline si è disposto di una coppia di Development Kit CY3274, prodotti dall'azienda Cypress, in grado ciascuno di supportare al meglio i chip CY8CPLC20 che vengono descritti in questo capitolo.

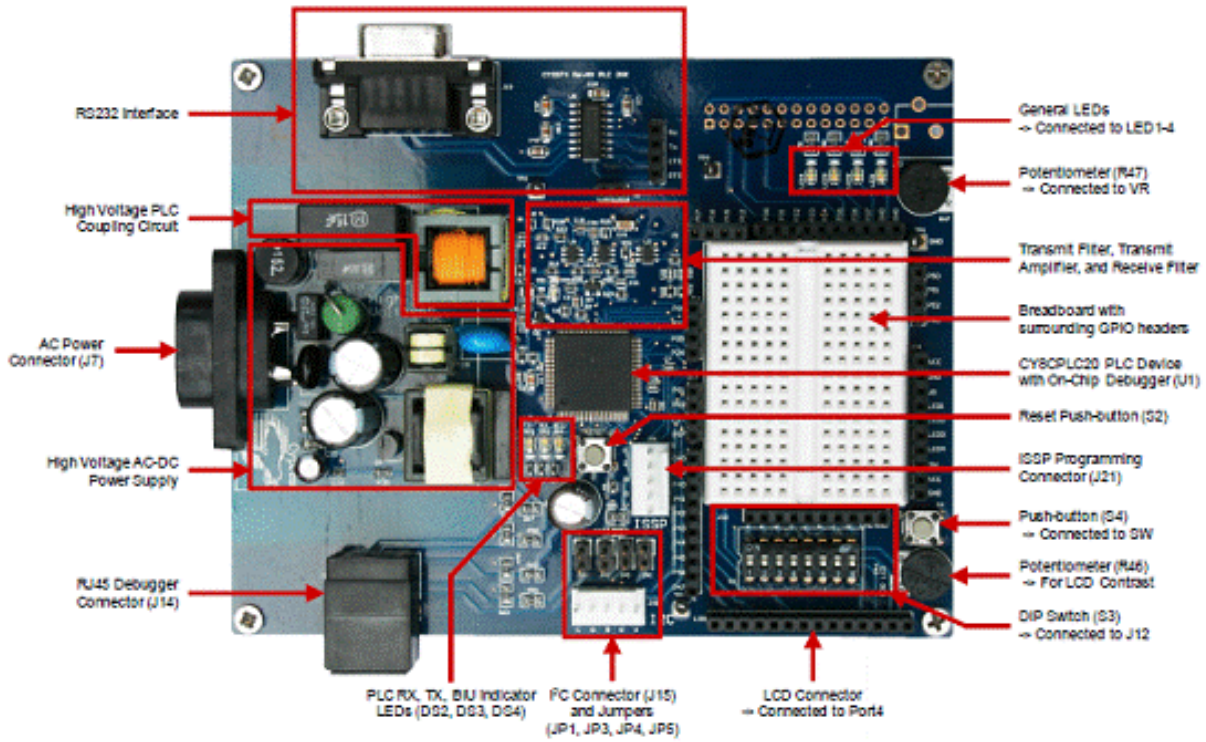


Figura 2.1: *Scheda del Development Kit CY3274 per il chip CY8PLC20*

Si tratta di chip dalle dimensioni e dai costi ridotti (14x14x1,4 mm, circa 6 euro al chip), programmabili mediante software proprietario, dotati di modem powerline PHY integrato (physical layer del modello ISO/OSI) per la trasmissione e la ricezione di segnali FSK (Frequency Shift Keying, nota codifica basata sul principio illustrato in Figura 2.2 dove sono indicate anche le reali frequenze utilizzate dal chip CY8CPLC20) ad una baud rate massima di 2400 bps.

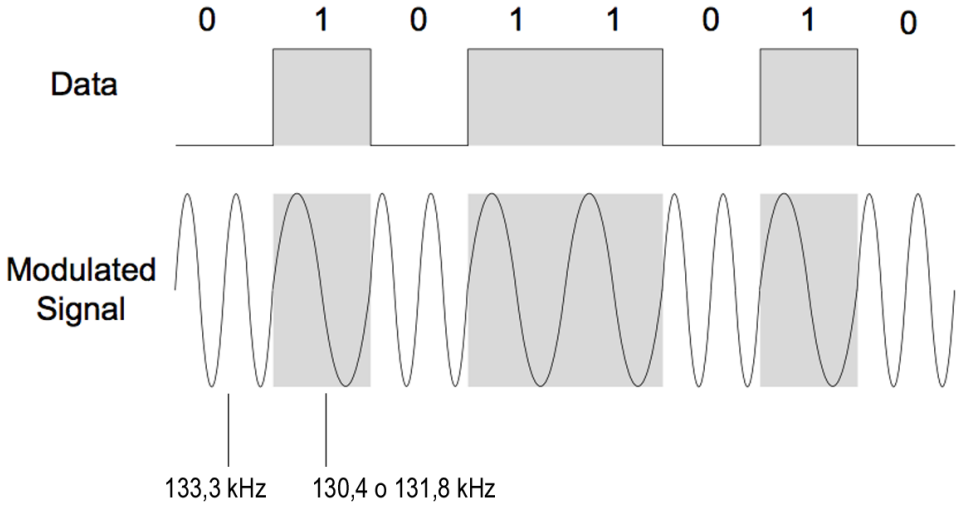


Figura 2.2: *Schema codifica FSK (Freqeuncy Shift Keying) - Un segnale portante viene modulato in frequenza dai dati che si vogliono trasmettere.*

Come sarà approfondito in seguito, da un punto di vista software i chip dispongono anche di un protocollo di rete il cui utilizzo è facoltativo in fase di programmazione, ottimizzato per l'indirizzamento singolo e multicast, l'utilizzo di comandi remoti, il controllo degli errori ed altre funzioni. Cypress propone anche un Application Layer I^2C (con opportuno socket per il collegamento hardware) di cui non si è usufruito in questo lavoro di tesi in quanto riduce il controllo sui registri. Per quanto riguarda il processore del chip si tratta di un M8C che lavora alla velocità massima di 24 MHz. La memoria interna è di 32 kB flash e 2kB SRAM. I chip comprendono inoltre una serie di convertitori DAC ed ADC e la scheda presenta molte periferiche di supporto come ad esempio 8 interruttori, 4 led, 4 jumper, un pulsante, un potenziometro, una presa ethernet, una seriale ed altre ancora. Per maggiori dettagli si veda la prima facciata della scheda tecnica del chip in Appendice, Figura A.1.

Caratteristica molto importante è invece la programmabilità dei pin I/O che, affiancata all'utilizzo dei PSoC Blocks in fase di programmazione, consente di utilizzare al meglio una moltitudine di moduli di vario tipo, come ad esempio dei Timer da 8 a 32 bit, un'interfaccia UART per il collegamento seriale al computer, un modulo per la gestione dell'LCD presente nella scheda del Kit e molti altri.

Per quanto detto, il chip comprende al suo interno sia il Physical Layer costituito dal modem FSK, che un protocollo di rete powerline; il circuito di accoppiamento con la rete invece si trova

nella scheda sul quale è installato il chip e Cypress ne fornisce gli schemi, in accordo con le normative CENELEC per l'Europa e FCC per il Nord America. Di seguito si può trovare un diagramma a blocchi del Physical Layer descritto in seguito.

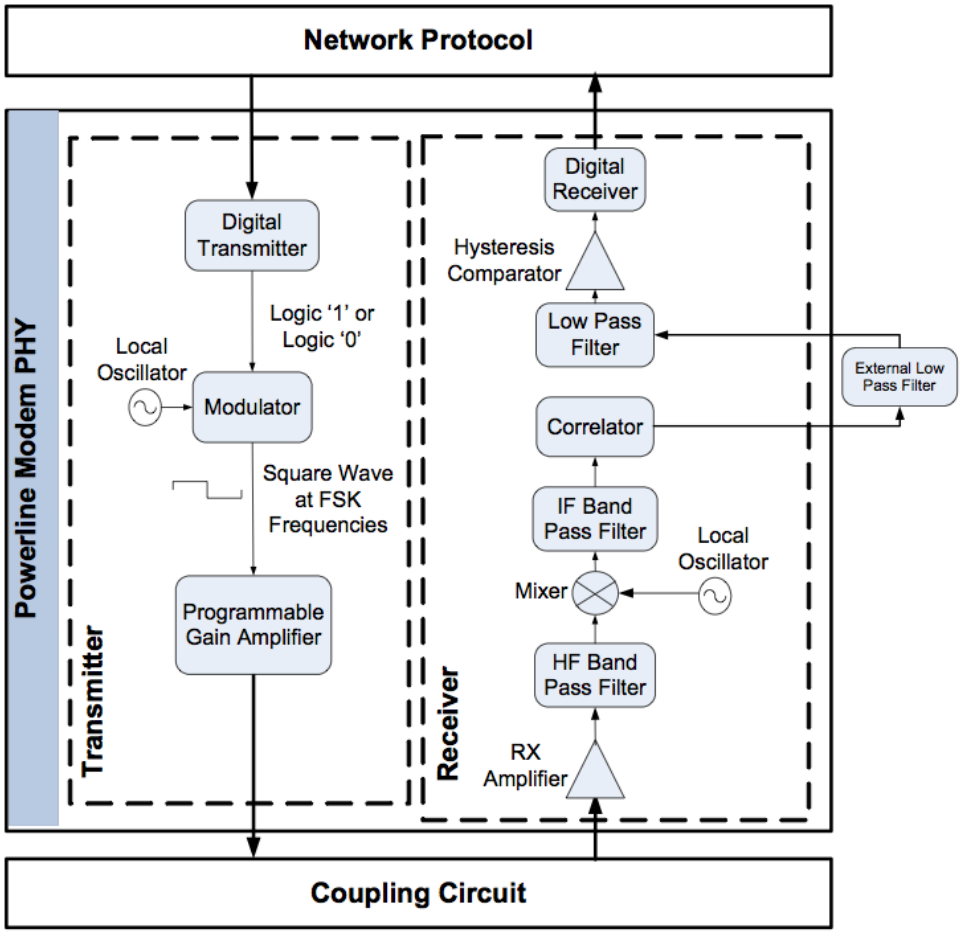


Figura 2.3: *Schema Physical Layer del Chip CY8PLC20*

Alcuni parametri del modem possono essere modificati mediante un apposito registro di sistema, l'$FSK\_Configuration\_Register$, che permette tra le altre cose di selezionare la velocità di trasmissione tra i valori 600, 1200, 1800 o 2400 bps. I dati digitali in arrivo dal network layer sono messi in serie e costituendo l'ingresso del modulatore determinano la frequenza dell'onda che verrà convogliata al segnale a 50Hz della rete. Il segnale di uscita è quindi un'onda quadra di frequenza 133.3 kHz per lo zero logico e di 131.8 kHz (oppure 130.4 kHz) per l'uno. L'ampiezza di questo segnale è regolabile tra 55 mV picco-picco e 3,5 volt picco-picco, mediante 4 bit del registro sopraccitato. Ciò permette di adattare il segnale trasmesso al rumore presente nella rete. Per quanto riguarda la ricezione invece, il segnale FSK in arrivo dal circuito di accoppiamento viene filtrato mediante un passa banda (spettro da 125 kHz a 140 kHZ) prima di essere demodulato mediante l'uso di un mixer (per moltiplicare il segnale ricevuto con un'altro

generato localmente), un correlatore, un passa basso ed un comparatore ad isteresi. In questo caso, mediante un apposito registro è possibile regolare la sensibilità del ricevitore modificando la soglia di RX Gain tra i valori 125 μVrms e 55 mVrms.

Cypress fornisce schemi per il circuito di accoppiamento per reti a 110 o 240 V alternata, in accordo con le normative FCC Part 15 per il Nord America e CENELEC EN50065-1:2001 per l'Europa. La programmazione prevede, come illustrato in seguito, anche la gestione di routine di interrupt programmate in assembly.

2.2 Il protocollo di rete Cypress

Il protocollo di rete fornito da Cypress comprende le funzioni di data link e network layer del modello ISO/OSI. Supporta molte caratteristiche tra le quali comunicazioni bidirezionali e master-slave, indirizzamento logico a 8-16 bit (indirizzo modificabile su ogni dispositivo), indirizzamento fisico a 64 bit, trasmissione individuale, broadcast o in group mode (ogni dispositivo può appartenere ad uno o più gruppi della stessa rete), accesso CSMA, trasmissione con conferma (Ack) o senza e totale controllo dei parametri relativi a tutte le funzioni citate. Tali funzionalità si riflettono sula struttura della Protocol Data Unit (PDU) che può contenere fino a 31 byte di dati.

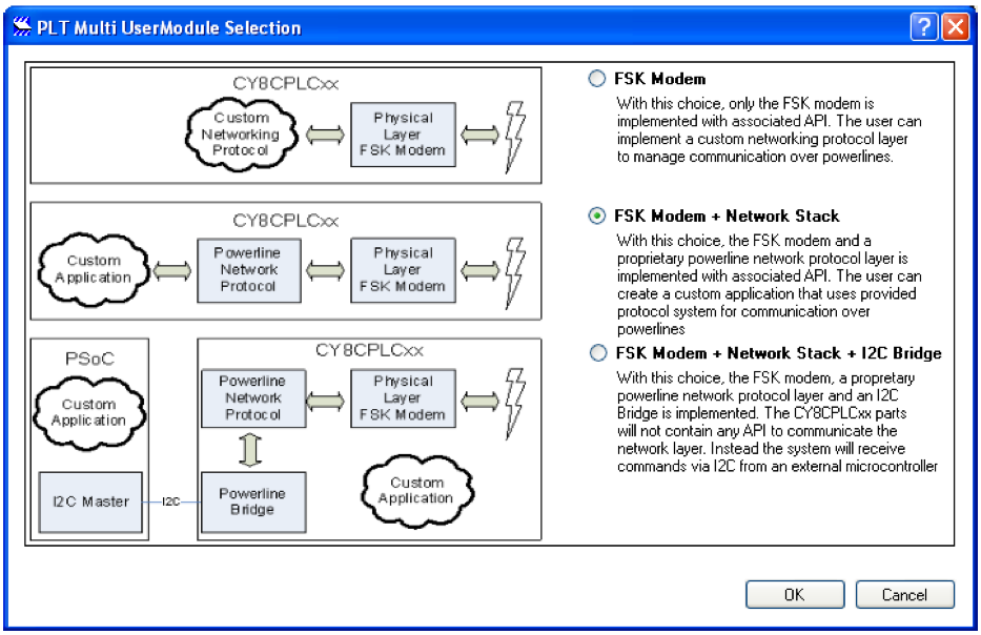


Figura 2.4: *Finestra per la selezione del Network Protocol in fase di programmazione*

L'accesso CSMA al mezzo è così gestito: ad ogni pacchetto inviato viene assegnato un tempo casuale tra 85 e 115 ms (con 7 valori casuali in questo range, parametri modificabili come descritto nella sezione registri). In questo intervallo il Band-In-Use (BIU) detector, ovvero un

sensore in grado di valutare se è già presente una trasmissione nella rete, deve stabilire che la rete non è in uso, altrimenti viene generato immediatamente l'interrupt di BIU Timeout oppure si hanno nuovi tentativi di accesso fino al sopraggiungere del BIUTimeOut dopo 1,1s/3,5s a seconda del rumore. Il Band-In-Use detector in particolare, come definito nella normativa CENELEC EN 50065-1, ritiene in uso la rete quando viene rilevato un segnale con una potenza superiore alla soglia di 86 dBmVrms (modificabile, trascurando la normativa) in qualsiasi punto nel range 131.5/133.5 kHz per più di 4 ms. Quando il trasmettitore va in time-out dopo un tempo compreso tra 1,1 e 3 secondi (dipende dal rumore nella rete) poiché la banda risulta in uso in tutto l'intervallo, viene generato un interrupt di fallimento dell'acquisizione della rete (si alza il flag $Status\_UnableToTX$ nel registro INT_Status, descritto in seguito).

Si analizza ora la struttura del pacchetto utilizzata dal Network Layer, osservabile in Figura 2.5, detto Powerline Transceiver (PLT) Packet.

<table>
<tr><th>Byte Offset</th><th colspan="8">Bit Offset</th></tr>
<tr><td></td><td>7</td><td colspan="2">6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td>0x00</td><td>SA Type</td><td colspan="2">DA Type</td><td>Service Type</td><td>RSVD</td><td>RSVD</td><td>Response</td><td>RSVD</td></tr>
<tr><td>0x01</td><td colspan="8">Destination Address
(8-Bit Logical, 16-Bit Extended Logical or 64-Bit Physical)</td></tr>
<tr><td>0x02</td><td colspan="8">Source Address
(8-Bit Logical, 16-Bit Extended Logical or 64-Bit Physical)</td></tr>
<tr><td>0x03</td><td colspan="8">Command</td></tr>
<tr><td>0x04</td><td colspan="3">RSVD</td><td colspan="5">Payload Length</td></tr>
<tr><td>0x05</td><td colspan="4">Seq Num</td><td colspan="4">Powerline Packet Header CRC</td></tr>
<tr><td>0x06</td><td colspan="8" rowspan="3">Payload (0 to 31 Bytes)</td></tr>
<tr><td></td></tr>
<tr><td></td></tr>
<tr><td></td><td colspan="8">Powerline Transceiver Packet CRC</td></tr>
</table>

Figura 2.5: *Struttura del PLT Packet utilizzato con il protocollo di rete Cypress*

La fase di costruzione del pacchetto è implementata internamente, tuttavia l'utente può modificare alcuni parametri manualmente agendo su determinati registri. L'invio nella rete di ogni pacchetto è preceduto da un byte di preambolo pari a "0xAB". I primi byte del pacchetto costituiscono un $header$ di lunghezza variabile (da 6 a 20) in base al tipo di indirizzamento utilizzato. La descrizione delle sigle che compaiono in Figura 2.5 è presentata in Figura 2.6.

Field Name	No. of Bits	Tag	Description
SA Type	1	Source Address Type	0 – Logical Addressing 1 – Physical Addressing
DA Type	2	Destination Address Type	00 – Logical Addressing 01 – Group Addressing 10 – Physical Addressing 11 – Invalid
Service Type	1		0 – Unacknowledged Messaging 1 – Acknowledged Messaging
Response	1	Response	0 - Not an acknowledgement or response packet 1 - Acknowledgement or response packet
Seq Num	4	Sequence Number	4-bit unique identifier for each packet between source and destination.
Header CRC	4		4-bit CRC value. This enables the receiver to suspend receiving the rest of the packet if its header is corrupted

Figura 2.6: *Descrizione parametri PLT Packet*

Appare chiaro come il tipo di indirizzamento influenzi la lunghezza dell'header e come sia possibile avere un payload variabile tra 0 e 31 byte (tutti i bit del Payload length a "1"). L'ultimo della struttura è un byte di controllo a ridondanza ciclica CRC. Appare chiaro che nel caso migliore, utilizzando il Network Layer, oltre ai dati che si vogliono trasmettere vengono trasmessi in rete minimo altri 8 byte tra $preambolo$, $header$ e crc. Il "Sequence number" viene utilizzato per la gestione dei retry e degli ack. Molto importante è il fatto che i byte nella rete vengono trasmessi secondo il protocollo della trasmissione seriale asincrona, ovvero con un bit di start, uno di parità ed uno di stop. Inoltre tra un byte e l'altro ci deve essere almeno un ciclo di durata Tbit=1/Vbps, pertanto nell'analisi delle prestazioni si terrà conto che ad ogni byte corrispondono 11 bit trasmessi in rete come mostrato in figura 2.7.

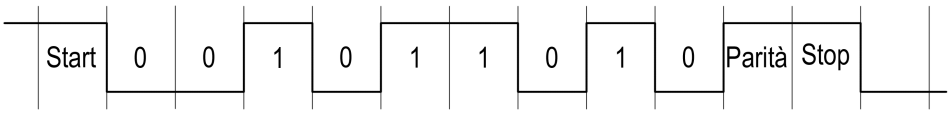


Figura 2.7: *Struttura UART della comunicazione powerline: 1 Bit Start, 8 Bit Dato, 1 Bit Parità, 1 Bit Stop*

Per quanto riguarda i tipi di indirizzamento si cita la possibilità di creare Single Group, ovvero assegnare ad ogni nodo uno ed un solo Single Group con indirizzo a 8 bit (256 Gruppi possibili); e quella di Multiple Group, assegnando ogni nodo ad uno o più dei primi 8 gruppi, attivando il corrispondente bit nell'apposito registro da un byte.
Il quarto byte dell'header contiene il comando da inviare: sono state infatti preimpostate alcune funzioni per modificare o richiedere il contenuto di alcuni particolari registri o far inviare i dati locali al richiedente. Nel dettaglio è possibile utilizzare uno dei seguenti comandi, codificati come osservabile in Appendice, Figura A.2. Quando il comando serve per modificare il valore di determinati registri, i nuovi contenuti vengono passati nel payload del pacchetto.

- $SetRemote\_TXEnable$, per modificare il TXEnable bit, descritto più avanti.
- $SetRemote\_ExtendedAddr$, per modificare la modalità di addressing
- $SetRemote\_LogicalAddr$, per modificare l'indirizzo logico vero e proprio
- $GetRemote\_LogicalAddr$, per ottenere l'indirizzo logico attuale
- $GetRemote\_PhysicalAddr$, per ottenere l'indirizzo fisico
- $GetRemote\_State$, per ottenere l'attuale $PLC\_ModeRegister$, descritto più avanti
- $GetRemote\_Version$, per conoscere il numero della versione del firmware
- $SendRemote\_Data$, comando di default per trasmettere dati al nodo
- $RequestRemote\_Data$, per richiedere l'invio dei dati locali da parte del nodo
- $ResponseRemote\_Data$, è il comando che si trova nel pacchetto di ritorno inviato dal nodo remoto dopo che abbia ricevuto il comando $RequestRemote\_Data$
- $SetRemote\_BIU$, per modificare l'uso del detector BIU
- $SetRemote\_ThresoldValue$, per modificare la soglia di BIU
- $SetRemote\_GroupMembership$, per modificare il tipo di appartenenza ad un gruppo tra Single Group e Multiple Group Mode.

2.3 I registri

Come accennato precedentemente, è possibile utilizzare o meno il protocollo di rete Cypress che comprende la gestione automatica dei pacchetti inviati e ricevuti. Quando questo protocollo non viene utilizzato, il numero di registri accessibili nella memoria si riduce notevolmente in quanto sta all'utente implementare il Network Layer. Una mappa completa dei registri presenti nella memoria è osservabile in Appendice, Figura A.3. In particolare, i registri accessibili in entrambe le modalità permettono la regolazione dei parametri relativi alle seguenti categorie di controlli.

Rumore nella rete e Band In Use Detection

- Registro 0x30, $Threshold\_noise$, contiene la soglia di BIU ($BIU\_Threshold\_Constant$) nei quattro bit meno significativi. Questa soglia coincide con il valore al di sotto del quale deve stare il rumore nella rete nelle frequenze di trasmissione affinché il chip determini che il mezzo trasmissivo sia libero. Tale soglia deve stare di poco sopra al rumore presente in rete e vi è la possibilità di utilizzare una funzione che lo imposti in automatico. Il valore selezionabile va da un minimo di 70 dBuVrms ad un massimo di 118 dBuVrms, con altri 14 valori distribuiti in modo irregolare in questo range. Il valore di default è di 87 dBVrms, in accordo con la normativa CENELEC. .

- Registro 0x34, $TimingConfig$, mediante il quale è possibile impostare i parametri dell'accesso CSMA. Due bit determinano il $BIU\_Interval\_Min$, ovvero il tempo minimo del range all'interno del quale viene estratto il tempo casuale di accesso al mezzo. I valori possibili sono 10, 20, 50 e 85 ms (quest'ultimo è il valore in accordo con la normativa CENELEC). Altri due bit compongono il $BIU\_Interval\_Span$, che coincide con l'ampiezza del range che può essere di 30ms o 15ms (entrambi con 7 valori casuali possibili, rigorosamente 30ms per sistemi CENELEC) o di 5ms (con un solo valore possibile). Il $BIU\_Timeout$ invece, costituito da un bit solo, permette di scegliere il tempo di Timeout del BIU detector di "1,1s" nel caso in cui non riuscisse ad acquisire la linea (con successivi ritentativi a tempi casuali) oppure di impostare il Timeout al primo rilevamento di BIU. In realtà quando si seleziona "1,1s", il tempo effettivo può arrivare fino a 3,5 ms e dipende dall'intensità del rumore nella rete. Infine gli ultimi 2 bit determinano l'$Ack\_Timeout$, ovvero il tempo di attesa massimo per la ricezione dell'Ack dopo la trasmissione di un dato prima di riprovare l'invio o di segnalare la mancata ricezione dell'Ack, in accordo con le impostazioni in vigore. Per quanto riguarda questo parametro i valori possibili sono "500 ms" fissi e tre valori del tipo "Auto+100ms", "Auto+50ms" e "Auto+20ms", dove Auto è un valore che dipende di volta in volta dalla lunghezza del pacchetto, dalla baud rate selezionata e dal TX_Delay impostato.

Modem FSK

- Registro 0x31, $Modem\_Config$, contiene le impostazioni del modem FSK. In particolare permette l'impostazione dei seguenti parametri:

  - $Modem\_TXDelay$, ovvero il ritardo fisso tra l'istruzione di invio dei dati e l'effettiva immissione nella rete degli stessi. Ciò da tempo alla circuiteria esterna di prepararsi. Questo valore può assumere i valori di 7, 13, 19 o 25 ms. Questa impostazione non riguarda soltanto l'invio dei dati da parte del dispositivo ma anche dell'ack di ritorno al momento della ricezione di un pacchetto (quando previsto). Questo tempo va ovviamente a penalizzare il tempo di trasmissione dei dati e non può essere annullato.
  - $Modem\_FSKBW$, ovvero la banda utilizzata dal modem. In particolare è possibile variare la frequenza di modulazione dell'uno logico, scegliendo tra 131,8 e 130,4 kHz, mentre lo zero logico rimane fisso a 133,3 kHz.

  - $Modem\_BPS$, che determina la velocità di trasmissione mediante 2 bit. I valori possibili sono 600, 1200, 1800 e 2400 bps.

- Registri 0x32 e 0x33, rispettivamente TX_Gain e RX_Gain. Il primo coincide con l'ampiezza della modulata per la trasmissione e va da 55mVp-p a 3,5Vp-p, con 14 valori distribuiti in modo irregolare nel range. Il secondo riguarda invece la sensibilità del sensore di ricezione e va da 125 μVrms a 5mVrms.

Indirizzo Fisico

- Registro 0x6A, $Local\_PA$, contiene un identificativo del chip, ha una lunghezza di 8 byte.

Versione Firmware

- Registro 0x72, $Local\_FW$, contiene la versione del firmware corrente.

Quando invece si utilizza il protocollo di rete sviluppato da Cypress, diventano accessibili anche molti altri registri che servono per sfruttare in vario modo le potenzialità del protocollo stesso. Di seguito una panoramica generale delle funzioni implementate in questa modalità d'uso, per una descrizione più dettagliata si rimanda ai data sheet del dispositivo. In particolare è possibile leggere e modificare i registri relativi a:

Interrupt

- Registro 0x00, INT_Enable, per l'abilitazione dei vari interrupt utilizzabili, uno per ogni bit del registro. Quando questi sono abilitati si ha la modifica del registro INT_Status e la chiamata della funzione $PLT\_HostInterrupt\_ISR$ situata nel file $PLT\_1INT.asm$ modificabile in fase di programmazione. Nel dettaglio:

  - INT_Clear, bit che va settato a zero dopo aver letto il registro dello stato degli interrupt quando uno di questi fosse occorso, serve per resettare lo stato.
  - $INT\_Polarity$, per selezionare la logica degli interrupt tra attivi alti e attivi bassi
  - $INT\_UnableToTX$, per attivare l'interrupt del timeout del BIU detector
  - $INT\_TX\_NO\_ACK$, per attivare l'interrupt che occorre in caso di mancata ricezione di ack laddove fosse invece prevista
  - $INT\_TX\_NO\_RESP$, abilita l'interrupt di nessuna risposta nel caso in cui fosse stato inviato il comando $Request\_Remote\_Data$ e non fosse stato ricevuto il pacchetto inviato dal dispositivo a cui è stato richiesto
  - $INT\_RX\_Packet\_Dropped$, abilita l'interrupt che occorre nel caso in cui un pacchetto venisse scartato a causa del buffer pieno
  - $INT\_RX\_Data\_Available$, abilita l'interrupt che segnala nuovi dati disponibili nel buffer

  - $INT\_TX\_Data\_Sent$, abilita l'interrupt che segnala il corretto invio di un pacchetto (con relativa ricezione dell'ack quando prevista)

- Registro 0x69, INT_Status, che comprende i flag dei vari interrupt (nel caso in cui questi fossero correttamente abilitati mediante il registro analizzato sopra). Il primo bit $Status\_Value\_Change$ indica una variazione nel registro stesso e può essere testato in polling per rilevare un avvenuto interrupt.

Indirizzamento

- Registri 0x01 e 0x02, $Local\_LA\_LSB$ e $Local\_LA\_MSB$, per l'indirizzamento logico. Sono byte modificabili, servono per la gestione mittente/destinatario nelle trasmissioni. Come accennato in precedenza è possibile scegliere se utilizzare l'indirizzamento logico ad uno (2^8 dispositivi nella rete) o a due byte (2^{16} dispositivi nella rete), da questo il significato delle sigle LSB e MSB.

- Registro 0x03, $Local\_Group$, per indicare il singolo gruppo di appartenenza del dispositivo. Mediante questo metodo è quindi possibile creare fino a 256 gruppi diversi nella rete per realizzare invii multicast.

- Registro 0x04, $Local\_Group\_Hot$, per indicare a quale o quali dei primi 8 gruppi appartiene il dispositivo in questione (0 non appartiene, 1 appartiene), oltre a quello indicato nel registro precedente.

- Registro 0x05, PLC_Mode, dove compaiono i seguenti bit:

  - TX_Enable, per abilitare l'invio dei dati, altrimenti possono essere inviati solo Ack
  - RX_Enable, per abilitare la ricezione dei dati, altrimenti possono essere ricevuti soltanto gli Ack
  - $Lock\_Configuration$, per consentire o meno la modifica delle configurazioni mediante accesso remoto (utilizzando i comandi analizzati in precedenza)
  - $Disable\_BIU$, per disattivare la BIU dettino prima dell'invio dei dati
  - $RX\_Override$, per stabilire cosa deve succedere in caso di ricezione di un pacchetto e buffer di ricezione pieno tra scartare quello nuovo oppure quello ancora nel buffer
  - $Set\_Ext\_Address$, per impostare l'utilizzo di 2 byte nell'indirizzamento logico
  - $Promiscuous\_MASK$, per decidere se scartare o meno i pacchetti ricevuti correttamente ma non destinati al dispositivo
  - $Promiscuous\_CRC\_MASK$, per impostare di non scartare i pacchetti ricevuti non correttamente, ovvero quelli che non superano il test CRC ma che sono destinati al dispositivo

Trasmissione

- Registro 0x06, $TX\_Message\_Length$, utilizzato principalmente per determinare la lunghezza del payload nel pacchetto da inviare nella prossima trasmissione. Come si può osservare in Appendice Figura A.3 è presente anche un bit $Send\_Message$ che viene utilizzato solo nella modalità Modem+Protocollo di rete+I^2C. In realtà in una delle applicazioni realizzate per questo lavoro di tesi si è utilizzato questo bit come flag per la gestione dal PC del comando di invio di un pacchetto via PLC (Capitolo 3).

- Registro 0x07, TX_Config, per la configurazione del tipo di indirizzamento utilizzato nel prossimo pacchetto sia per il mittente (TX_SA_Type: logico e fisico) che per il destinatario (TX_DA_Type: logico, fisico o gruppo), per il tipo di messaggio ($TX\_Service\_Type$: con o senza Ack di ritorno) e per il numero di invii/tentativi di invio. Nel dettaglio infatti, se si utilizza un servizio con Ack, il Tx_Retry indica il numero di tentativi di invio nel caso in cui non si ricevesse l'Ack relativo all'invio precedente; se invece si usa un servizio senza ack, il TX_Retry indica il numero di invii effettuati ad ogni comando di trasmissione.

- Registro 0x08, TX_DA, per l'indirizzo del destinatario o del gruppo (1, 2 o 8 byte a seconda del tipo di indirizzamento)

- Registro 0x10, $TX\_CommandID$, per il comando che si vuole inviare o per specificare l'invio di dati, in accordo con quanto visto in precedenza.

- Registro 0x11, TX_Data, per i dati da inviare, massimo 31 byte.

Ricezione

- Registro 0x40, $RX\_Message\_INFO$, in cui compaiono:

  - New_RX_Msg, un bit di flag che diventa alto in caso di ricezione di un nuovo messaggio e va abbassato dopo la lettura per resettare il registro status
  - RX_DA_Type, per il tipo di indirizzamento utilizzato per il destinatario nel messaggio ricevuto
  - RX_SA_Type, per il tipo di indirizzamento utilizzato per il mittente
  - RX_Msg_Length, per la lunghezza del payload nel pacchetto ricevuto

- Registri 0x41, 0x49 e 0x4a, RX_SA, $RX\_CommandID$, RX_Data, analoghi a quelli di trasmissione visti precedentemente

2.4 La programmazione

La programmazione avviene ad alto livello mediante un software Cypress, PSoC Designer. Si tratta di un IDE molto intuitivo basato sul Drag&Drop e sui linguaggi C ed Assembly. Per poter gestire a pieno una periferica collegata alla scheda, è sufficiente trascinare il relativo modulo desiderato dalla lista User Modules al diagramma a blocchi del chip che si sta programmando. Immediatamente si potrà disporre, in fase di programmazione, di tutte le funzioni relative a quel modulo; ad esempio, dopo aver incluso il modulo LCD nel programma, si potrà usare una funzione come $LCD\_PrCstring(\text{"}testo\text{"})$ per mostrare la scritta $testo$ sull'LCD. Per ogni modulo è disponibile in rete un datasheet che ne descrive in maniera completa il funzionamento, le caratteristiche e le API utilizzabili in programmazione sia C che assembly.

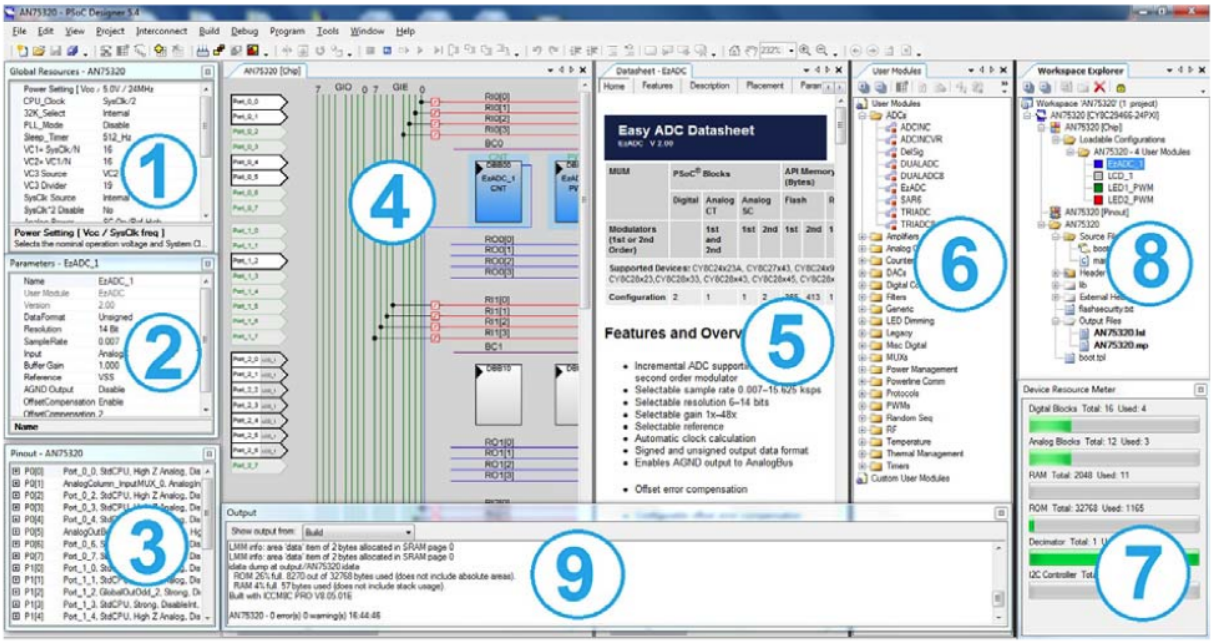


Figura 2.8: *Software PSoC Deisgner*

In figura 2.8 è possibile osservare come si presenta l'interfaccia di PSoC Designer dove le aree numerate indicano rispettivamente:

- 1) Risorse Globali, CPU, impostazioni dei clock, ecc.
- 2) Parametri del modulo selezionato
- 3) Piedinatura del modulo selezionato
- 4) Chip-level editor, diagramma a blocchi del chip in programmazione
- 5) Datasheet

- 6) User Modules, libreria dei moduli utente utilizzabili
- 7) Device Resource Meter, strumento di misura delle risorse del dispositivo con la configurazione attuale
- 8) Workspace, cartelle del progetto attivo
- 9) Output.

La programmazione principale avviene in C e consiste nella realizzazione del file main.c, sorgente di un programma che una vota compilato e caricato nel chip viene eseguito una volta ad ogni reset della scheda. In assembly è possibile programmare principalmente le routine di interrupt: PSoC Designer permette infatti l'assegnazione di singoli interrupt ad una grande vastità di eventi che si possono verificare, sia software che hardware. Dopo aver compilato il progetto ed aver verificato che non ci siano errori, mediante un dispositivo chiamato MiniProg (Figura 2.9) che viene collegato ad uno specifico socket della scheda ed al PC mediante USB, è possibile caricare il programma nel chip (l'operazione richiede circa 30 secondi). Dopo aver staccato il MiniProg si preme il pulsante di reset della scheda che avvia il nuovo programma.

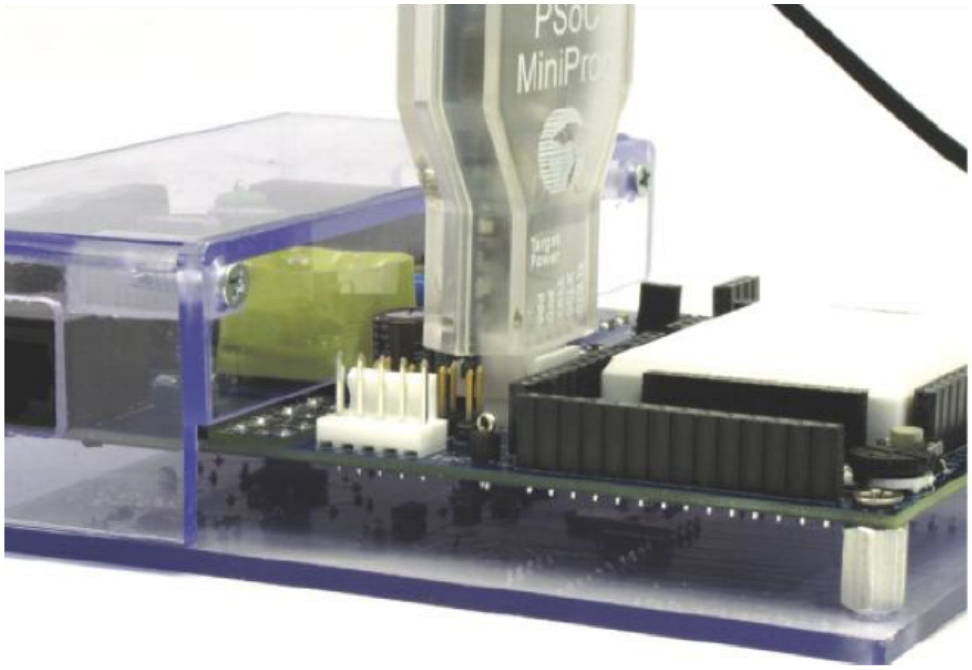


Figura 2.9: *MiniProg per la programmazione del chip*

Per quanto riguarda la programmazione specifica del chip powerline, si utilizza il modulo utente PLT che, una volta trascinato nel chip-editor, richiede se si vuole utilizzare o meno il protocollo di rete Cypress. Si dispone così immediatamente delle funzioni relative alla trasmissione e ricezione di dati mediante PLC. Nel dettaglio, le funzioni utilizzabili indipendentemente dall'uso del protocollo di rete sono le seguenti:

- PLT_Start, per inizializzare il chip all'utilizzo come transceiver
- PLT_Stop, per disabilitare tutti i blocchi digitali ed analogici relativi al modem FSK
- $PLT\_Restart$, per riavviare i blocchi PLT mantenendo il contenuto dei registri (situati nel cosiddetto PLT_Memory_Array)
- $PLT\_DisableInt$, per disabilitare tutti gli interrupt del PLT
- $PLT\_EnableInt$, per abilitare tutti gli interrupt del PLT
- $PLT\_AutoSetBIUThreshold$, per impostare automaticamente la BIUThresholdConstant (soglia di rumore oltre il quale la linea viene determinata in uso) ad un valore subito al di sopra del rumore rilevato attualmente nella rete.

Le funzioni utilizzabili solamente senza il protocollo di rete invece sono:

- $PLT\_SwitchToReceiver$, per utilizzare il modem come ricevitore
- $PLT\_SwitchToTransmitter$, per utilizzarlo come trasmettitore. Appare chiaro che senza il protocollo di rete non è possibile utilizzare contemporaneamente entrambe le funzioni.
- $PLT\_SendData$, per inviare i dati nella powerline, da 0 a 255 bytes.
- $PLT\_AcquirePowerline$, per verificare lo stato di BIU attuale
- $PLT\_RXEnableInt$ e $PLT\_RXDisableInt$, per abilitare o disabilitare gli interrupt di ricezione (va a modificare i bit del registro INT_Enable)
- $PLT\_bReadRXStatus$, per ottenere il contenuto del Receiver Control Register e verificare la disponibilità o meno di dati ricevuti nel buffer.
- $PLT\_bReadRxData$, per ottenere l'ultimo byte ricevuto

Infine le funzioni utilizzabili solo con il protocollo di rete:

- $PLT\_SendMsg$, peri inviare un pacchetto. La sua struttura è determinata dalla configurazione attuale dei registri visti in precedenza.
- PLT_Poll, funzione da eseguire ciclicamente sia per verificare la disponibilità di nuovi byte ricevuti e non perderne eventuali altri in arrivo a causa del buffer pieno, che per dare inizio all'invio di un pacchetto e testare il registro NT_Status in polling per conoscerne l'esito della trasmissione.

Gli interrupt software (chiamati così anche se in realtà l'esecuzione del programma non prosegue) che si possono gestire sono i seguenti se non si utilizza il protocollo di rete:

- $PLT\_BIU\_Active\_ISR$, in caso di rilevamento BIU positivo,

- $PLT\_BIU\_Complete\_ISR$, quando si ha un rilevamento BIU, all'inizio del rilevamento successivo viene lanciato questo interrupt
- $PLT\_TX\_Active\_ISR$, quando il trasmettitore inizia a trasmettere nella powerline
- $PLT\_TX\_Complete\_ISR$, quando la trasmissione è completata

Se invece si utilizza il protocollo di rete gli interrupt software sono i seguenti:

- $PLT\_HostInterrupt\_ISR$, per una qualsiasi delle 6 condizioni descritte ed abilitate dal registro INT_Enable
- $PLT\_RX\_Active\_ISR$, quando si ha l'inizio di ricezione di un nuovo byte dalla powerline
- $PLT\_RX\_Complete\_ISR$, quando il byte è stato ricevuto completamente

Quando non si usa il protocollo di rete infine è possibile gestire anche un interrupt derivante dall'hardware, ovvero:

- PLT_DIG2_ISR, quando un nuovo byte è ricevuto dalla powerline. Consiste nell'interrupt abilitato e disabilitato con i comandi $PLT\_RXEnableInt$ e $PLT\_RXDisableInt$.

Capitolo 3

Strumentazione utilizzata per l'analisi delle prestazioni

3.1 Configurazione e utilizzo development kit

3.1.1 Panoramica Sofware e Hardware

Per poter eseguire i test di funzionamento si sono sviluppati alcuni programmi. In particolare:

- per misurare l'effettiva velocità di trasmissione senza utilizzare il protocollo di rete si è realizzato un programma per il chip in grado di rilevare il tempo necessario per l'invio di un numero variabile di byte e comunicarlo mediante l'LCD della scheda

- per analizzare i tempi di trasmissione dei pacchetti utilizzando il protocollo di rete ed il $packet\ error\ rate$ invece, si è scelto di realizzare un programma per i chip ed una relativa applicazione Windows per il controllo e la gestione dei chip da PC via cavo seriale RS232 e per il salvataggio automatico dei log di ogni trasmissione avvenuta; uno script di matlab ricava successivamente le statistiche di interesse dai risultati ottenuti.

- per valutare le percentuali di errore in una routine di polling è stata realizzato un altro programma per i chip che sfrutta la stessa interfaccia windows del punto precedente

Da un punto di vista hardware la scheda è stata impostata allo stesso modo per tutte le applicazioni, tuttavia non tutti i collegamenti effettuati sono stati sempre sfruttati.

In Figura 3.1 si può osservare la scheda così come viene utilizzata.

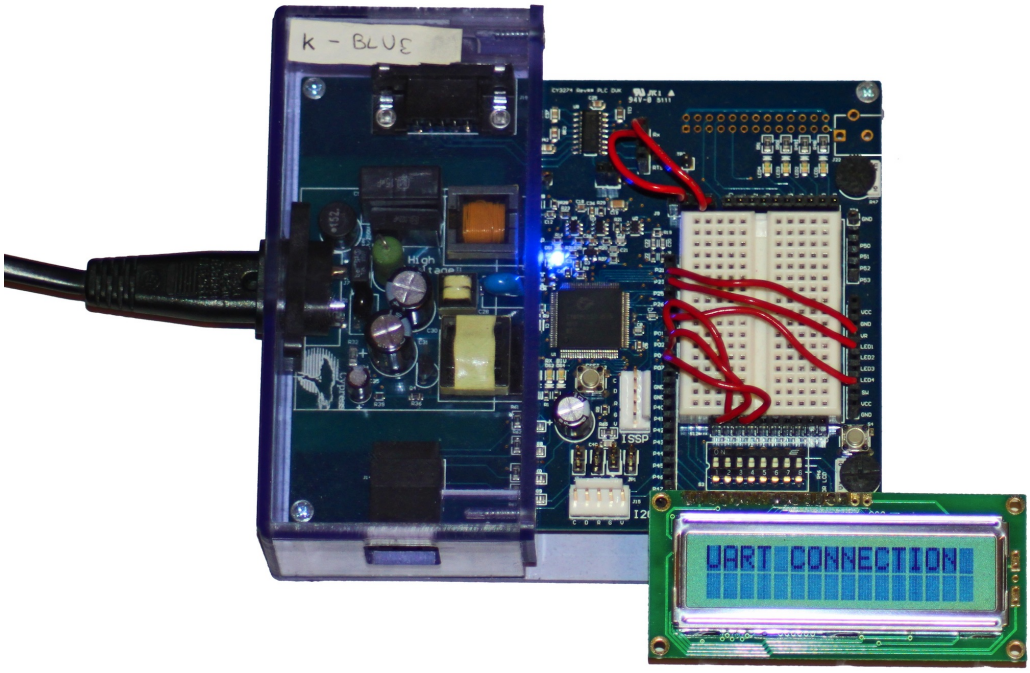


Figura 3.1: *Configurazione hardware della scheda*

In particolare si sono eseguiti i seguenti collegamenti:

- collegamento LCD nell'apposito socket a 14 piedini
- collegamento del pin P01, digital I/O, al pulsante SW della scheda
- collegamento dei pin P02, P04 e P07, digital I/O, a tre interruttori diversi della scheda, rispettivamente al secondo, al primo ed al terzo
- collegamento dei pin P21, P23 e P25 a tre LED della scheda, rispettivamente LED1, LED2 e LED3.
- collegamento dei pin P16 e P12 ai pin Rx e Tx della porta seriale per l'interfacciamento UART

Nell'immagine si può notare anche il socket del MiniProg, il dispositivo fornito con il Development Kit che serve per trasferire il programma nel chip.

Nei paragrafi seguenti vengono descritti i programmi menzionati.

3.1.2 Software chip senza protocollo di rete

In PSoC Designer, il software proprietario per la programmazione dei chip, sono stati utilizzati i seguenti Moduli Utente:

- BIU_LED, TX_LED e RX_LED per la segnalazione di BIU detection, trasmissione o ricezione in corso mediante i tre led
- PLT, PowerLine Transceiver, impostato senza l'uso del protocollo di rete
- LCD, per poter mostrare i risultati ottenuti mediante l'LCD della scheda
- $TIMER16$, un contatore interno a 16 bit per le misure dei tempi

Per quanto riguarda l'uso dei **LED** è sufficiente configurare correttamente i parametri del modulo ovvero la porta alla quale viene connesso (in accordo con i collegamenti fatti) e la logica utilizzata (attivo acceso o spento). Dopo di questo si procede configurando adeguatamente il pin utilizzato, assegnando un nome (uguale al nome del modulo) ed un drive (tra $High\ Z$, $PullUp$, $PullDown$, $Strong$; per farlo funzionare con il LED si seleziona questa ultima voce come descritto nel documento AN54416 fornito da Cypress). I LED sono utilizzati per segnalare il rilevamento di BIU, la ricezione e la trasmissione in corso, pertanto in fase di programmazione vengono aggiunte le righe di codice Assembly $lcall\_LED\_On$ e $lcall\ _LED\_Off$ nelle opportune routine di interrupt nel file $PLT\_1INT.asm$. In particolare si devono accendere quando si attivano gli interrupt $PLT\_BIU\_Active\_ISR$, $PLT\_TX\_Active\_ISR$ e $PLT\_RX\_Active\_ISR$. Si devono invece spegnere negli interrupt $PLT\_BIU\_Complete\_ISR$, $PLT\_TX\_Complete\_ISR$ e $PLT\_RX\_Complete\_ISR$.

L'**LCD** non necessita alcun settaggio particolare poichè il collegamento può essere effettuato solamente nel socket dedicato. Una volta aggiunto il modulo al progetto è possibile utilizzare le seguenti istruzioni:

- $LCD\_Start()$, per avviare il display
- $LCD\_Position(r, c)$, per posizionare il cursore di scrittura nella casella della riga r e colonna c. L'LCD ha in totale 2 righe e 16 colonne e la prima posizione in alto a destra è numerata come (0,0).
- $LCD\_PrCString("testo")$, per stampare una stringa sull'LCD a partire dalla posizione attuale verso destra.
- $LCD\_PrHexByte(byte)$ per stampare un byte in formato esadecimale

Per la configurazione degli **interruttori** è sufficiente assegnare un nome al rispettivo Pin utilizzato ed impostarlo in modalità $PullUp$. In fase di programmazione è possibile ottenere la posizione dell'interruttore testando il bit del registro del rispettivo port. Ad esempio, il registro del $Port0$, al quale sono collegati entrambi gli interruttori, si chiama $PRT0DR$.

Il **pulsante**, analogamente, viene configurato come $PullDown$ e viene testato nello stesso identico modo degli interruttori.

Il **Timer16** invece, una volta aggiunto al progetto, deve essere appositamente configurato. In particolare vanno impostati il Periodo ed il clock. Per l'utilizzo si è consultato il datasheet relativo al Modulo in questione. Si è scelto di collegarlo al clock generato dalla CPU di 32 kHz. In questo modo è possibile misurare un intervallo massimo di circa 2 secondi utilizzando 2 byte. In particolare, ad ogni ciclo di questo clock a 32 kHz, il registro viene decrementato di uno; con due byte è quindi possibile ottenere un tempo massimo di:

$$T_{max} = Tclock \cdot 2^{16} = \frac{1}{32kHz} \cdot 65536 = 2,048\,s. \tag{3.1}$$

Per controllarlo in fase di programmazione si possono utilizzare le seguenti funzioni:

- $Timer\_Start()$, per farlo partire, ricordando che conta alla rovescia
- $Timer\_Stop()$, per stopparlo
- $TX\_Timer\_wReadTimer()$, per conoscere il valore attuale del contatore
- $TX\_Timer\_WritePeriod(Period)$, per assegnare un valore al contatore in genere per resettarlo.

Come accennato il periodo iniziale utilizzato è $0xFFFF$ ed essendo una grandezza a 16 bit, i valori letti vengono assegnati a variabili di tipo word. L'intervallo misurato si ottiene come differenza tra il valore iniziale ed il valore misurato mediante $Timer\_wReadTimer()$. Per convertirlo in secondi è necessario moltiplicare questo valore per il periodo del clock, in pratica dividerlo per 32000. Per avere un valore preciso si è scelto di mostrare sul display del chip il valore in esadecimale per poi convertirlo e dividerlo con un calcolatore, senza dover affrontare il problema della stampa sul display di un valore con la virgola.

Di seguito si ha uno schema a blocchi del programma.

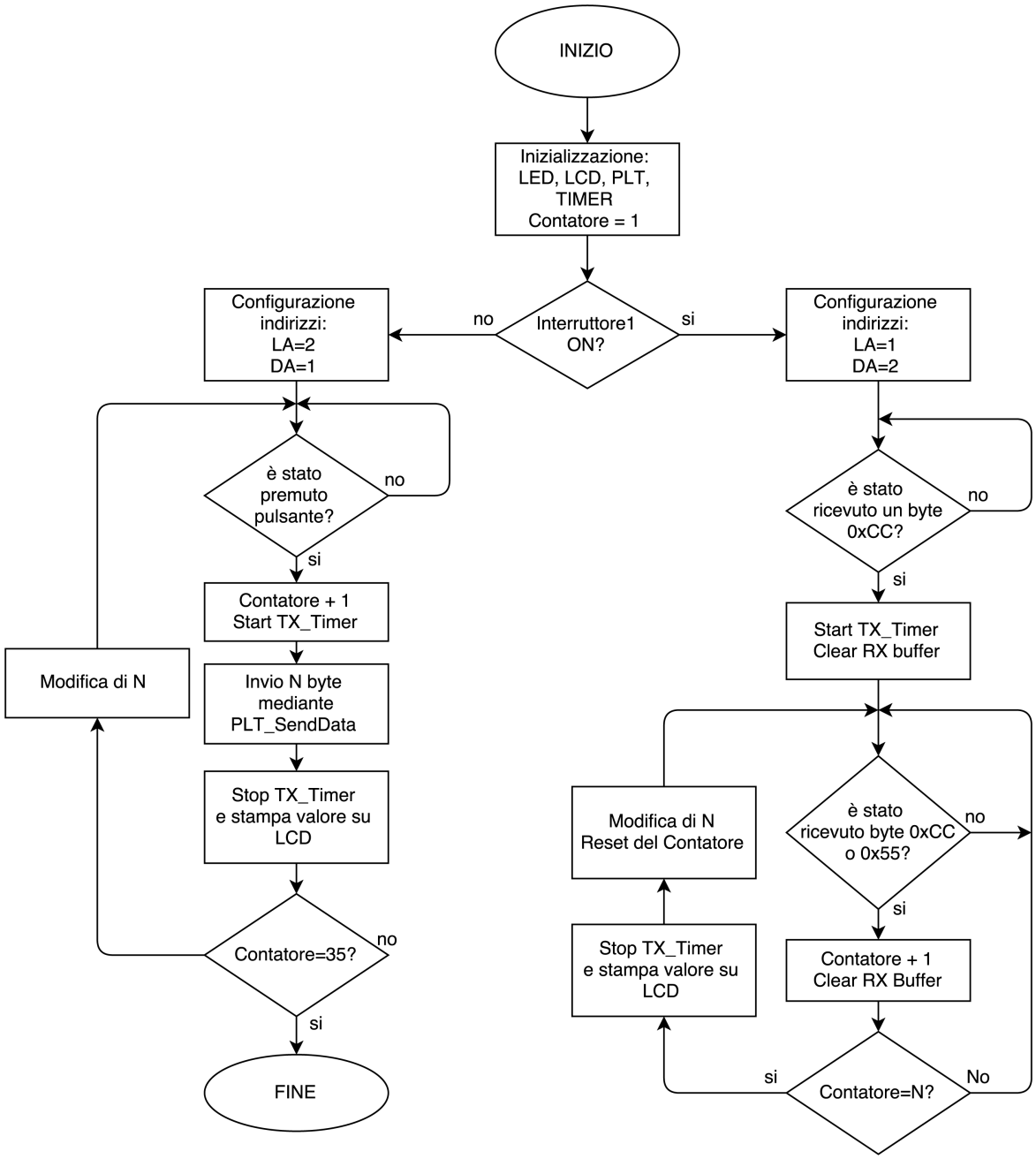


Figura 3.2: *Schema a blocchi della struttura principale del programma senza protocollo di rete Cypress*

Prima di avviare il programma vero e proprio si ha una sezione di inizializzazione in cui vengono abilitati gli interrupt globabli del chip, inizializzato il modulo PLT, avviati i LED e l'LCD: infine mediante l'interruttore 1, chiamato ADD_Select e connesso alla porta P04, è possibile selezionare l'indirizzo logico del dispositivo che determina anche se lo stesso verrà utilizzato come ricevitore o come trasmettitore, per semplificare la programmazione. A questo punto si possono differenziare la parte di programma relativa al primo dei due da quella relativa al secondo.

Per quanto riguarda il **trasmettitore**, appena viene premuto il pulsante di start (collegato alla porta P01), si è scelto di inviare una serie di dati, avviando il timer prima di iniziare la trasmissione e fermandolo quando questa si conclude. Per eseguire più prove senza riprogrammare il chip si è scelto di variare la lunghezza in byte della serie dinamicamente ed automaticamente per poter così inviare complessivamente 5 serie da 250 byte, 5 da 200, 5 da 150, 5 da 100, 5 da 50, 5 da 10 e 5 da 1.

In caso di **ricevitore** invece, il programma inizia testando in loop il registro dello stato della ricezione mediante l'apposita istruzione $PLT\_bReadRxStatus$, in particolare valutando il bit relativo al "buffer pieno" che si ha quando un byte è stato ricevuto. Quando ciò si verifica il programma procede verificando che il byte ricevuto sia uno di quelli inviati e non una falsa trasmissione dovuta al rumore nella rete. Non disponendo di un controllo di rete, si è scelto di inviare una serie di byte alternati del tipo 0xCC-0x55 e di verificare che in ricezione si abbiano solo questi tipi di byte. Finché non viene ricevuto un byte 0xCC il ricevitore continua a rimanere nel loop di attesa. Quando viene ricevuto 0xCC, il Timer viene avviato ed il ricevitore inizia a contare il totale di byte corretti (0xCC o 0x55) ricevuti. Al raggiungimento del numero di byte prestabiliti viene fermato il timer e letto il valore, che rappresenta quindi il tempo totale per la ricezione degli (N-1) byte inviati. A questo punto si procede con il reset del contatore e la modifica del numero di byte della prossima serie (che dipende dal numero di serie ricevute fino ad ora). Ad ogni ricezione si è scelto inoltre di mostrare sull'LCD gli ultimi 8 byte ricevuti. Grazie a questo, una volta disabilitato il controllo sul byte ricevuto (ammettendo quindi anche byte diversi da 0xCC o 0x55) si è potuto osservare che in caso di rumore nella rete e guadagno di ricezione (RX_Gain) troppo basso, il dispositivo riceve falsi dati, spesso uguali ad 0xFF o valori che differiscono da questo per uno o più bit.

Nel programma compare anche una funzione utilizzata per stampare sull'LCD il valore decimale contenuto in una variabile di uno o due byte per poter mostrare il valore dei contatori e talvolta quello dei tempi (senza eseguire la conversione esadecimale-decimale manualmente, perdendo però la parte decimale dei valori con la virgola).

Le valutazioni su come ricavare la velocità di trasmissione in bps a a partire dal tempo misurato vengono esposte nel capitolo seguente.

3.1.3 Software chip con protocollo di rete

Come accennato in precedenza, per questa fase dei test sono stati realizzati un programma per il chip in PSoC Designer ed uno per PC Windows che interagiscono mediante connessione seriale. Nella programmazione del chip sono stati usati i seguenti moduli, alcuni dei quali già presentati nel paragrafo precedente e utilizzati allo stesso modo quando non specificato:

- BIU_LED, TX_LED e RX_LED
- PLT, PowerLine Transceiver, impostato con l'uso del protocollo di rete
- LCD
- $TIMER16$
- $PWM8\_UART$ e $UART$, due moduli necessari per l'utilizzo della comunicazione via cavo seriale RS-232.

In particolare il **PWM8** è un Pulse Width Modulator ad 8 bit e serve per fornire un clock adatto al modulo $UART$ che invece si occupa della serializzazione dei dati. Si è scelto di trasferire i dati a 9600 bps, pertanto il PWM deve generare un clock a 9600*8=76,8 kHz, poiché ogni bit trasmetto o ricevuto necessita di 8 impulsi di clock. Per configurare il PWM è necessario selezionare il clock di ingresso ed i vari divisori dei quali l'ultimo è (Period + 1)=25+1=26, come descritto nel datasheet del modulo UART. Utilizzando come clock di ingresso il VC2, settato come VC1/6 che a sua volta è pari al SysClk/2, si ha:

$$f_{UART} = ((SysClk/2)/6)/26 = 76,9\,kHz. \tag{3.2}$$

Per connettere il PWM al modulo UART si usa uno dei BUS, in particolare il $Row\_2\_Output\_0$. Infine si impostano adeguatamente i Pin di Tx ed Rx utilizzando rispettivamente i bus $Row\_2\_Output\_2$ e $Row\_2\_Input\_2$, utilizzati in modalità driver $Strong$. A questo punto si utilizza un protocollo di comunicazione che prevede tre tipi di pacchetti di cui i primi due possono essere inviati dall'Host (il PC) mentre l'ultimo dal chip:

- "**UART Write Packet**", per scrivere dati nel PLT_Memory_Array del chip
- "**UART Read Packet**", per richiedere la comunicazioni di dati presenti nel PLT_Memory_Array del chip
- "**UART Response Packet**", per ricevere dati dal chip

Il primo serve per scrivere un determinato numero (Length) di byte (Dato) nel PLT_Memory_Array a partire da un certo indirizzo (Offset) in poi ed ha la struttura di figura 3.3

	7	6	5	4	3	2	1	0
Byte 0	0	Length						
Byte 1	Offset							
Byte 2+	Data							

Figura 3.3: *Struttura del "Write Packet" per la scrittura dei registri da cavo seriale*

Il secondo serve per la lettura di un dterminato numero (Lenght) di byte dal PLT_Memory_Array a partire da un certo indirizzo (Offset) in poi ed il pacchetto ha la struttura di figura 3.4

	7	6	5	4	3	2	1	0
Byte 0	0	Length						
Byte 1	Offset							
Byte 2+	Data							

Figura 3.4: *Struttura del "Read Packet" per la lettura dei registri da cavo seriale*

Il terzo invece viene utilizzato per la semplice comunicazione al PC dei dati richiesti al chip ed ha la seguente struttura.

	7	6	5	4	3	2	1	0
Byte 0+	Data							

Figura 3.5: *Struttura del "Response Packet" per la comunicazione dei registri via cavo seriale*

Sul chip viene inserita una routine (interrupt handler) che, in caso di interrupt dovuto alla ricezione di dati via UART, analizza il pacchetto ricevuto e lo scrive nella memoria oppure risponde inviando dati richiesti. Ad ogni interrupt $Host\_UART\_ISR$, chiamato alla ricezione di un byte via UART, viene eseguita una particolare porzione di codice della interrupt handler dipendentemente dalla posizione del byte nella serie che si sta analizzando. E' presente anche un sistema che permette di stabilire quando un byte ricevuto è il primo di una nuova serie o appartiene ancora ad una serie precedente; per farlo si sfrutta un timeout calcolato mediante un conteggio di cicli del PWM. Un byte a 9600 bps viene ricevuto circa ogni 1.25 ms; si è stabilito un timeout di 2.6 ms, misurato con 200 cicli del PWM utilizzando un contatore aumentato ad ogni interrupt PWM_UART_ISR ($\frac{1}{f_{PWM}} \cdot 200 = 2,6\,ms$). Il diagramma a blocchi della routine interrupt handler è osservabile in Figura 3.6. Si fa notare che per accedere al byte ricevuto si utilizza la locazione $UART\_RX\_BUFFER\_REG$.

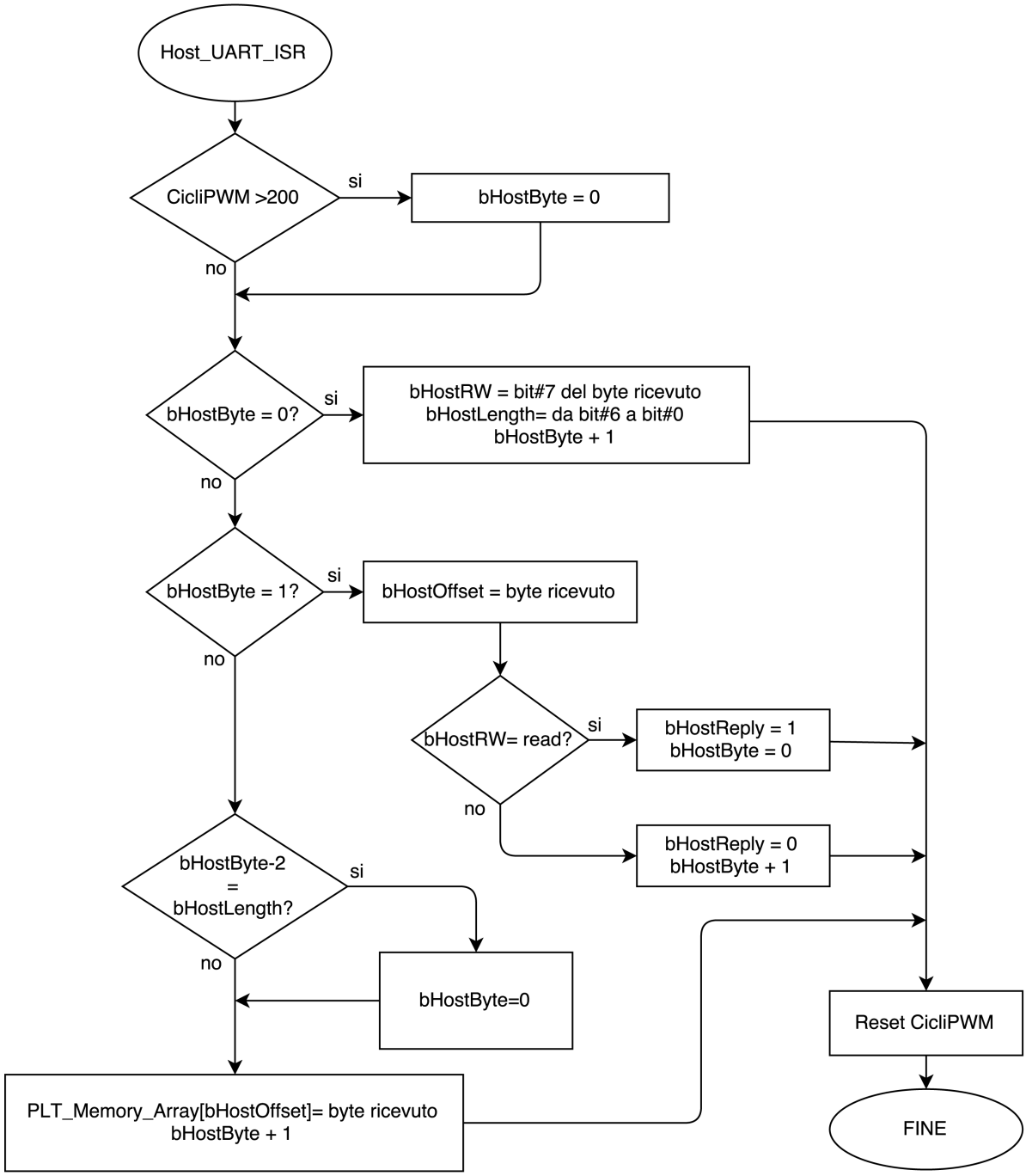


Figura 3.6: *Schema a blocchi della routine di ricezione byte via UART nel chip*

A questo punto è possibile analizzare il codice del file main.c di questo programma. Come detto nel capitolo 2, utilizzando il protocollo di rete Cypress i dispositivi possono funzionare contemporaneamente da trasmettitori e da ricevitori, in pratica, agendo sempre da ricevitori ec-

cetto durante l'invio dei dati. Il primo interruttore viene comunque utilizzato per la selezione dell'indirizzo logico in modo da poter caricare lo stesso identico programma in entrambi i chip. Dopo l'inizializzazione dei vari componenti già visti, vengono inizializzati anche il PWM ed il modulo UART, selezionando la struttura della trasmissione senza bit di parità. In questo modo la struttura della trasmissione nel cavo seriale è osservabile in Figura 3.7.

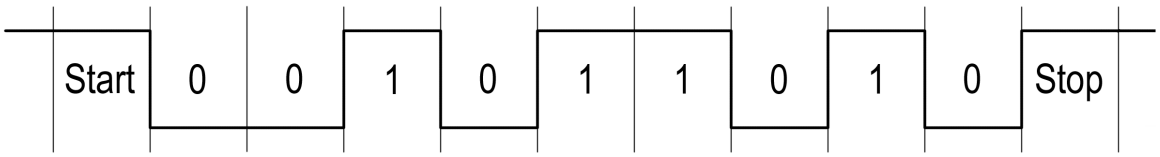


Figura 3.7: *Struttura della comunicazione UART su cavo seriale*

Vengono poi assegnati dei valori iniziali anche a tutti i registri del PLT_Memory_Array relativi al protocollo di rete Cypress, in particolare si impostano:

- la frequenza del modem FSK a 133,3/130,4 kHz
- la velocità di 2400 bps
- $TXEnable$ ed $RXEnable$ abilitati
- $Disable\_BIU$ (per non avere l'accesso CSMA alla rete, eliminando aleatorietà nei tempi di trasmissione)
- indirizzamento logico singolo a 1 byte
- trasmissione con Acknowledgement
- TXGain a 1,55 Vpp
- RXGain a 250 μVrms
- lunghezza payload a 0 byte
- abilitazione di tutti gli interrupt del PLT
- comando del pacchetto inviato: trasmissione dati $CMD\_SENDMSG$

La parte principale del programma (schema a blocchi in Figura 3.9) consiste in un loop continuo in cui vengono testati ad ogni ciclo uno per volta i seguenti bit:

A) il bit $Status\_RX\_Data\_Available$ del registro INT_Status che segnala la ricezione di un pacchetto via PLC

B) il bit relativo alla posizione del secondo interruttore, utilizzato per abilitare la funzione di rilevamento intensità del rumore nella rete

C) il bit relativo alla posizione del terzo interruttore, utilizzato per abilitare la trasmissione continua di 5000 pacchetti di dimensioni diverse via PLC, con conseguente invio via UART dei dati relativi ad ogni singola trasmissione

D) il bit $Send\_Message$ del registro $Message\_Length$, utilizzato come accennato in precedenza, per poter controllare dal PC via UART l'invio di un pacchetto nella powerline

E) il bit di flag $bHostReply$ che segnala di aver ricevuto una richiesta di lettura dati via UART e di dover procedere quindi con l'invio dei dati richiesti

Come accennato prima la ricezione di un pacchetto via UART comporta la scrittura o la lettura di un certo numero di registri della memoria. La differenza tra le due sta nel primo bit del primo byte ricevuto. Quando è richiesta la scrittura, la routine stessa, dalla ricezione del terzo byte compreso in poi, copia ogni byte ricevuto nell'indirizzo di memoria descritto dal secondo byte ricevuto, incrementando ovviamente l'indirizzo della memoria di ciclo in ciclo. Quando invece è richiesta la lettura, la routine di gestione dell'interrupt UART abilita il flag $bHost\_Reply$, punto E della lista precedente, che segnala la necessità di inviare via UART i dati richiesti. Se dal PC viene inviato un comando che va a modificare il bit $Send\_Message$ del registro $Message\_Length$ (punto D della lista precedente), il programma sul chip provvede ad inviare un pacchetto via PLC e a resettare il bit $Send\_Message$. Ogni volta che il chip invia un pacchetto via PLC lo fa mediante un'opportuna funzione chiamata $PLC\_Transmit\_Packet$ che integra la gestione del Timer per poter determinare e memorizzare ad ogni invio il tempo impiegato (dettagli in seguito). Quando viene rilevato un pacchetto ricevuto via PLC, punto A della lista precedente, il programma del chip prosegue aumentando il contatore di pacchetti ricevuti, stampando questo numero in decimale sull'LCD e azzerando il flag New_RX_Msg che automaticamente resetta il registro INT_Status e predispone il chip ad una nuova ricezione. Quando il secondo interruttore determina attiva la modalità di misurazione del rumore nella rete, punto B della lista precedente, il chip esegue l'istruzione $PLT\_AutoSetBIUThreshold$ che come detto in precedenza imposta il valore della soglia di BIU ad un valore minimamente superiore a quello rilevato attualmente nella rete. Leggendo poi il valore appena impostato è ottenibile una stima per eccesso del rumore nella rete. Commentando o meno alcune istruzioni in questa sezione di codice è possibile decidere se comunicare il rumore misurato via LCD o via UART. Quando invece viene determinato l'invio dei 5000 pacchetti mediante il terzo interruttore della scheda, punto C della lista precedente, il programma compie 5000 cicli all'interno di ognuno dei quali, oltre ad inviare un pacchetto mediante la funzione $PLC\_Transmit\_Packet$, invia tramite UART i dati relativi alla trasmissione. In particolare si è scelto di inviare al PC un pacchetto con una struttura ben definita, osservabile in Figura 3.8, contenente parte del log della trasmissione.

PACCHETTO UART LOG DI TRASMISSIONE		
Numero byte	Contenuto	Valore
Byte 1	PREAMBOLO	0xAA
Byte 2	PREAMBOLO	0xBB
Byte 3	ESITO	0xE1(Successo) o 0xE2 (Fallimento)
Byte 4	TEMPO LSB	0xXX
Byte 5	TEMPO MSB	0xXX
Byte 6	RUMORE	da 0x00 (70 dBuVrms)a 0x0F (118 dbuVrms)
Byte 7	LUNGHEZZA PAYLOAD	da 0x00 (0 byte) a 0x1F (31 byte)
Byte 8	STOP	0xBB
Byte 9	STOP	0xAA

Figura 3.8: *Struttura del pacchetto UART di Transmission Log contenente informazioni sulla trasmissione PLC appena avvenuta*

Come si può osservare sono presenti:

- due byte di start, 0xAA e 0xBB
- un byte per l'esito, successo 0xE1 o fallimento 0xE0
- due byte per il tempo di trasmissione, iniziando dal meno significativo
- uno per il rumore rilevato nella rete
- uno per la lunghezza del payload del pacchetto PLC inviato
- due byte di stop, 0xBB e 0xAA.

La conversione del tempo e del rumore viene affidata al programma Windows mentre l'analisi dei dati viene effettuata mediante appositi script MATLAB.
Mediante questo metodo si è scelto di inviare:

- 1000 pacchetti da 0 byte
- 1000 pacchetti da 7 byte
- 1000 pacchetti da 15 byte
- 1000 pacchetti da 23 byte
- 1000 pacchetti da 31 byte

La funzione $PLC\_Transmit\_Packet$ è stata strutturata in questo modo: inizialmente viene eseguita l'istruzione $PLT\_AutoSet\_BIUThreshold$ per adattare la soglia di BIU al rumore nella rete ed automaticamente ottenere, come già accennato, la stima del rumore attuale. Successivaente viene eseguito il comando $PLT\_SendMsg()$ ed avviato il Timer. A questo punto mediante un loop continuo il programma testa il registro INT_Status. Nel caso in cui andasse a uno il

bit $Status\_UnableToTX$, ovvero il chip non fosse in grado di acquisire la linea e fosse raggiunto il timeout di 1,1 secondi, la funzione proseguirebbe con l'innalzamento della soglia di BIU, un nuovo invio ed il restart del timer. Quando invece va ad 1 uno dei bit $Status\_TX\_Data\_Sent$, $Status\_TX\_NO\_ACK$ o $Status\_TX\_NO\_RESP$, il timer viene stoppato ed il tempo viene memorizzato. Solo nel caso di $Status\_TX\_Data\_Sent$ la funzione restituisce 1, indicando l'esito positivo della trasmissione. Si ricorda che questo bit va ad 1 alla ricezione dell'ACK quando previsto, altrimenti al termine dell'invio, mentre $Status\_TX\_NO\_RESP$ viene utilizzato solamente in combinazione con il comando remoto di invio dati. Uno schema a blocchi del programma è osservabile in Figura 3.9, mentre in Figura 3.10 viene mostrato lo schema a blocchi della funzione $PLT\_Transmit\_Packet()$.

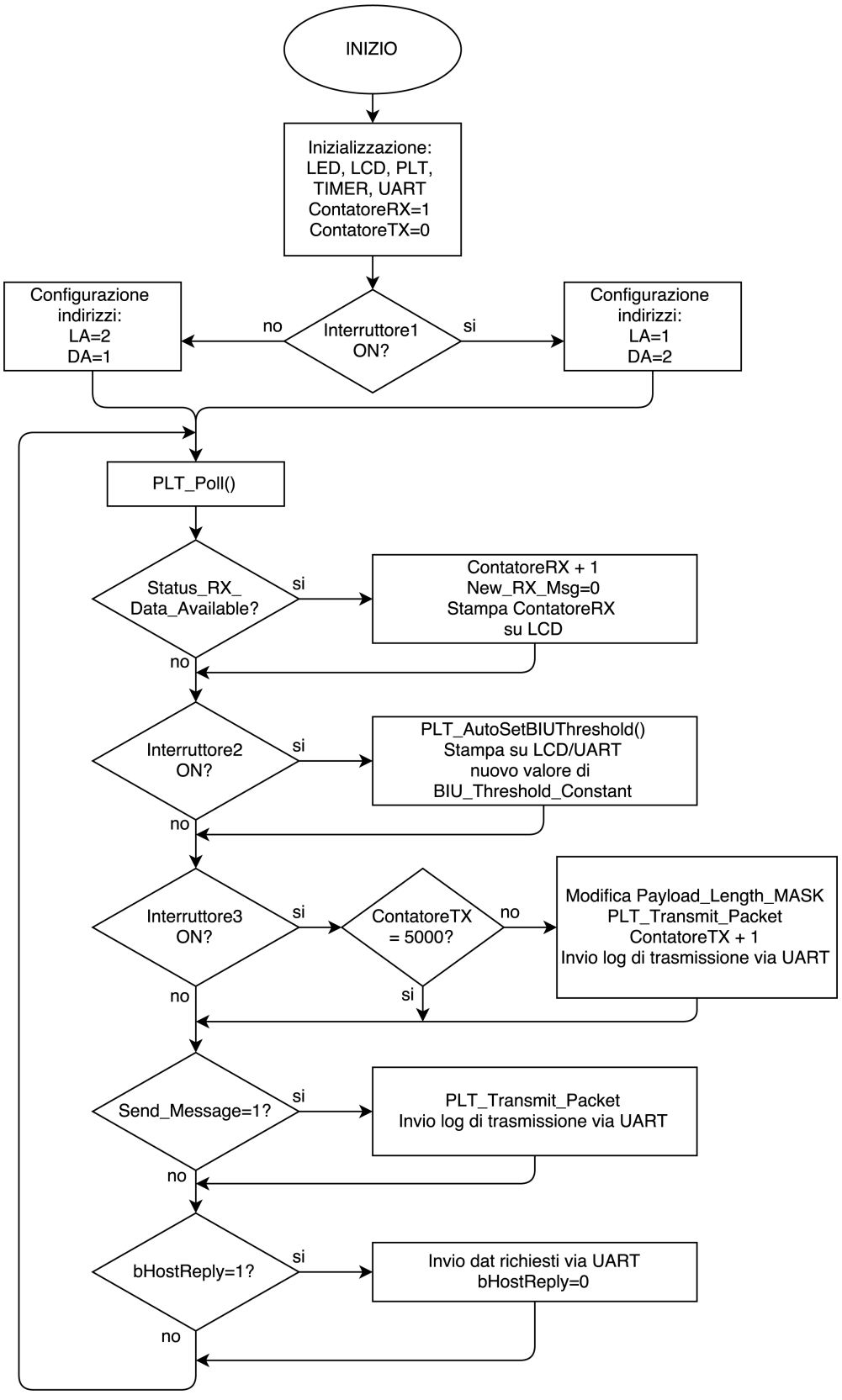


Figura 3.9: *Schema a blocchi del programma con il protocollo di rete Cypress*

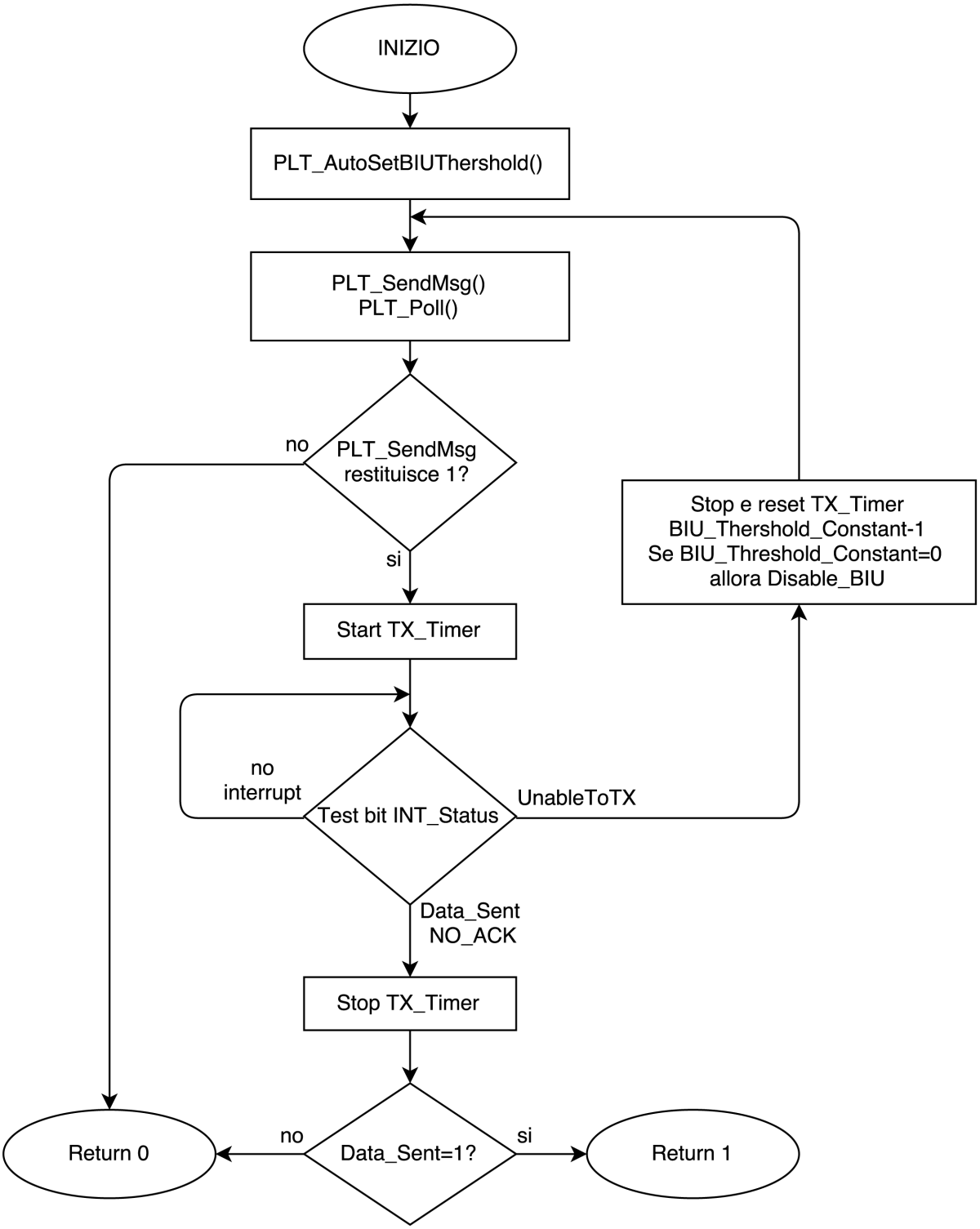


Figura 3.10: *Schema a blocchi della funzione* $PLT\_Transmit\_Packet()$

3.1.4 Software Windows per gestione chip con cavo seriale

L'applicazione Windows è stata realizzata con Visual Studio. Consiste in una Form unica nella quale sono presenti controlli di vario tipo che permettono di inviare automaticamente i comandi corretti via UART al chip e di gestirne le risposte. Prima di procedere con l'utilizzo del programma è necessario installare i driver del cavo seriale utilizzato e conoscere la porta COM ad esso relativa.

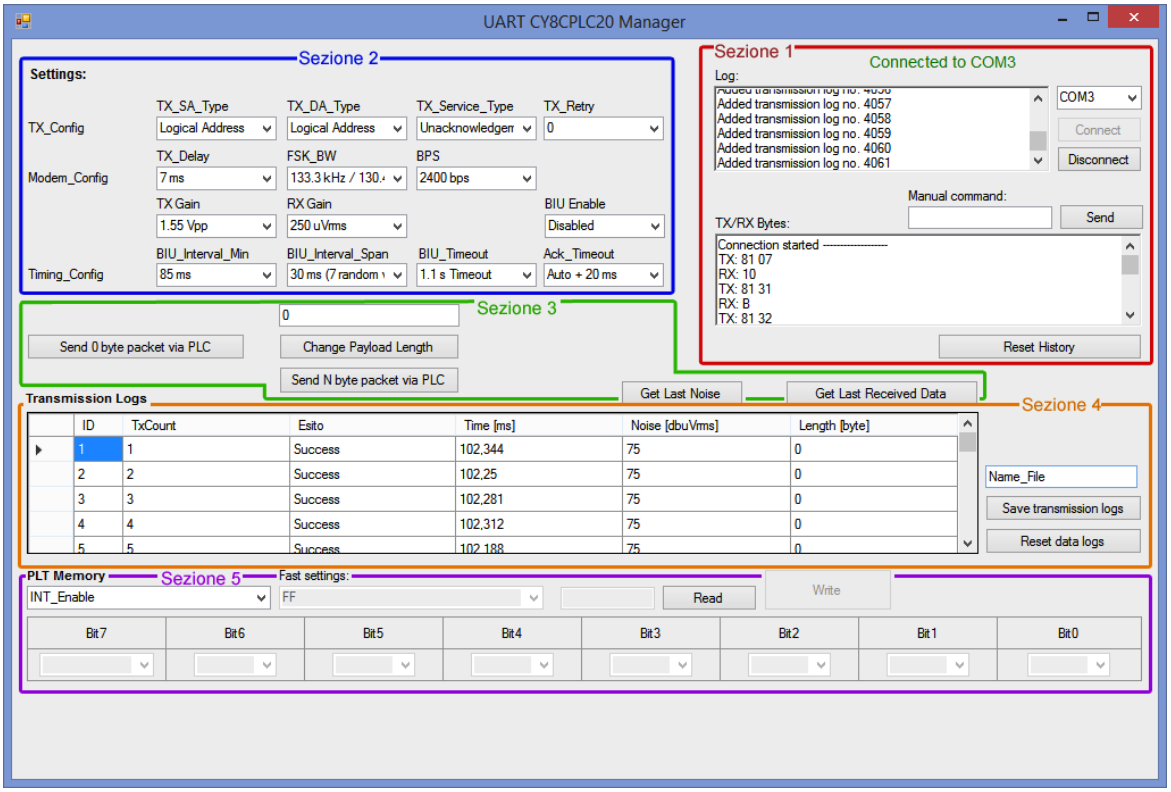


Figura 3.11: *Applicazione Windows per il controllo del chip da PC*

L'interfaccia grafica è quella di figura 3.11. Come si può osservare si individuano 5 sezioni principali:

Sezione 1 connessione/disconnessione, finestre di log, comandi manuali

Sezione 2 impostazioni del chip

Sezione 3 comando remoto di invio pacchetto via PLC, acquisizione rumore e ultimi dati ricevuti

Sezione 4 acquisizione e salvataggio log di trasmissione

Sezione 5 lettura/scrittura di qualsiasi registro della memoria

Mediante la **Sezione 1** $Connection$, dopo aver selezionato la porta COM corretta, è possibile premere il pulsante $Connect$ per avviare la comunicazione. In caso di connessione riuscita la scritta rossa "Not Connected" viene sostituita con "Connected to COM10" (ad esempio) di color verde. Nella casella di testo "Log" viene tenuta traccia degli eventi accaduti, come ad esempio la variazione di impostazioni, la ricezione di un log di trasmissione, l'invio di un comando manuale, la connessione e la disconnessione. Nella casella di testo "TX/RX Bytes" invece vengono scritti singolarmente in formato esadecimale tutti i byte inviati e tutti quelli ricevuti. Per comodità è previsto un pulsante di reset per queste due caselle. Tra le due finestre vi è lo strumento "Manual command" che permette di inviare pacchetti del tipo "Write Packet" o "Read Packet" visti in precedenza direttamente in formato esadecimale. Il programma provvede a tradurli da testo a byte esadecimali ed inviarli via UART al chip.

Al momento della connessione il programma invia automaticamente una serie di richieste di dati al chip per conoscere le attuali impostazioni dei registri e selezionare automaticamente le voci corrette nelle tendine della **Sezione 2** $Settings$. Da questa è possibile poi modificare rapidamente le impostazioni dei registri TX_Config, $Modem\_Config$, $Timing\_Config$, TX_Gain, RX_Gain e BIU_Enable. Quando l'utente seleziona una specifica voce di un menu a tendina, il programma costruisce automaticamente e spedisce il pacchetto "Write Packet" di 3 byte (struttura presentata in Figura 3.3). Come descritto nel paragrafo precedente, il programma presente nel chip provvede a scrivere il valore ricevuto come terzo byte nella locazione di memoria indicata dal secondo. Intanto il programma Windows invia la richiesta di lettura del registro appena scritto per avere conferma del successo della scrittura.

Sotto alla sezione $Settings$ è posizionata la **Sezione 3**, dove si trovano dei pulsanti per comandare l'invio di dati via PLC sfruttando, come descritto nel paragrafo precedente, il flag $Send\_Message$ del registro $Message\_Length$. In pratica:

- mediante il pulsante "Send 0 byte packet via PLC" il programma invia il "Write Packet 0x01 0x06 0x80", per la scrittura ("0x01") del byte "0x80" nel registro di indirizzo "0x06"; questo è l'indirizzo del registro $TX\_Message\_Length$ e il valore assegnato comporta la scrittura della lunghezza del pacchetto, rappresentata dai primi 5 bit, e quella del bit $Send\_Message$. Analogamente quando si utilizza il pulsante "Send N byte packet via PLC" il programma determina con che byte sostituire il terzo inviato (0x80) in modo da assegnare il valore corretto ai bit che determinano la lunghezza del payload. Il pulsante "Change Payload Length" invece provvede solo a modificare i primi 5 bit del registro $TX\_Message\_Lenght$, senza abilitare il flag di invio

- mediante "Get Last Noise" si può ottenere l'ultima soglia di rumore utilizzata dal BIU detector

- mediante "Get Last Received Data" è possibile osservare sulla finestra "Logs" le caratteristiche ed il contenuto dell'ultimo pacchetto ricevuto

La **Sezione 4** $Transmission\ Logs$ invece consiste in una tabella, inizialmente vuota, in cui sono presenti le seguenti colonne visibili:

- ID, contatore locale del programma Windows,
- $TxCount$, contatore presente nel chip, va da 0 a 255 e cicla
- $Esito$, "Success" o "Fail" in base al relativo byte ricevuto via UART
- $Time$, in millisecondi con la virgola, calcolato dopo aver unito i due relativi byte ricevuti via UART
- $Noise$, valore in dBVrms corrispondente al relativo byte ricevuto via UART
- $Length$, in numero di byte, corrispondente al relativo byte ricevuto via UART

e le seguenti colonne non visibili: TX_Gain, RX_Gain, $BIU\_Interval\_Min$, $BIU\_Interval\_Span$, $BIU\_Timeout$, $Ack\_Timeout$, BIU_Enable, $Service\_Type$ e TX_Retry. Quando il chip effettua una trasmissione PLC, come detto in precedenza, viene inviato via UART un pacchetto di log che presenta due byte di preambolo. Quando, in qualsiasi momento, vengono ricevuti questi due byte, il programma entra in una modalità speciale di "ricezione log di trasmissione via UART" ed i byte successivi vengono elaborati per completare le colonne visibili della tabella appena descritta. Per il riempimento di quelle invisibili invece vengono salvate le attuali impostazioni del chip, mostrate nella **Sezione 1**, acquisite precedentemente. Al termine della ricezione del pacchetto via UART, il programma esce dalla modalità di ricezione in cui era entrato, aggiunge la riga appena costruita alla tabella e lo comunica nella finestra "Logs". Quando si utilizza la procedura di invio di 5000 pacchetti nel chip, la tabella viene completata quasi in Real-Time man mano che vengono effettuate le trasmissioni PLC. In qualsiasi momento è possibile salvare tutti i log ricevuti utilizzando il campo "Name File" ed il pulsante "Save transmission logs". La tabella viene salvata in formato $.csv$ (comma separated values), un formato di testo che può essere facilmente importato in MATLAB o qualsiasi altro programma di elaborazione dati. Al nome del file il programma aggiunge automaticamente la data, l'ora e lo stato di abilitazione del BIU per una individuazione più veloce del contenuto dei vari file di log. Infine un pulsante di $Reset$ permette di svuotare completamente la tabella ed azzerare il contatore.

L'ultima, la **Sezione 5**, chiamata $PLT\ Memory$, consiste in un menu a tendina dal quale è possibile selezionare per nome uno qualsiasi dei registri della memoria del chip. Automaticamente il programma invia al chip il comando UART per la lettura del registro in questione e al momento della ricezione completa la serie di combobox "True o False" relative ad ognuno dei bit del registro dei quali viene adeguatamente modificato il nome. Viene inoltre scritto il valore esadecimale contenuto nel registro nella casella di testo affianco al pulsante $Read$. Quando possibile, ad esempio per il registro $TX\_CommandID$, nel secondo menu a tendina compaiono i nomi dei comandi impostabili e in caso di selezione di uno di questi, le 8 tendine dei bit vengono modificate automaticamente ed è possibile scrivere il registro utilizzando il pulsante $Write$.

Analogamente, anche completando la casella di testo affianco al pulsante $Read$ scrivendo un valore esadecimale, le combobox si sistemano automaticamente.

In fase di programmazione si sono anche gestite:

- Le inizializzazioni della serial port per una trasmissione senza bit di parità
- Le abilitazioni di tutti i comandi solo in caso di connessione stabilita
- L'invio all'inizio della connessione dei comandi necessari alla lettura dei registri di settings alternato alla relativa attesa della risposta e conseguente modifica dei menu della sezione 1. In questa sezione alcuni registri vanno a determinare lo stato di più menu a tendina, fatto di cui bisogna tenere conto sia alla ricezione dello stato, che all'invio dei comandi al chip per modificare solo i bit relativi ad un menu a tendina senza modificare gli altri.
- Routine di ricezione byte, in grado di rilevare automaticamente i pacchetti di transmission log inviati dal chip
- Routine di completamento tabella transmission log con elaborazione dei byte ricevuti. Per il tempo di trasmissione vengono uniti due byte ricevuti separatamente in un unica variabile tradotta poi in un valore in millisecondi.
- Funzione in grado di tradurre stringhe di byte esadecimali in variabili realmente esadecimali ed inviarle via UART
- Funzione di salvataggio file $.csv$
- Funzione per determinare valore esadecimale corrispondente alla posizione di 8 combobox relative ai bit e viceversa
- Struttura di variabili personalizzata per la memorizzazione di tutti i registri della memoria con relativi nomi dei bit e valori preimpostati assegnabili
- Gestione errori dell'utente, come ad esempio completamenti di caselle di testo in modo non ammesso e simili

3.1.5 Software chip per test di polling con protocollo di rete

Si è scelto di modificare il programma del chip precedente in modo che realizzasse una comunicazione di tipo polling. Nel chip trasmettitore ad ogni invio di pacchetto, dopo aver avviato il $Timer16$, dopo aver testato il registro status, si è scelto di attendere in un loop la ricezione di un nuovo pacchetto e di fermare il tempo quando ciò accade. E' stato anche previsto un timeout d'attesa di 1 secondo nel caso la trasmissione del pacchetto inviato o di quello di risposta non andasse a buon fine. Quando si prevede l'utilizzo con accesso CSMA e TX_Retry maggiore di zero il valore di 1 secondo potrebbe non bastare. Nel chip ricevitore invece sono state inserite apposite istruzioni nella routine di ricezione di un nuovo messaggio: mediante test sul registro $RX\_Message\_INFO$ il ricevitore determina la lunghezza del payload nel pacchetto ricevuto e ne invia immediatamente uno della stessa dimensione. Anche con questo sistema si è scelto di comandare l'invio di 5000 pacchetti di payload variabile progressivamente. Lo schema a blocchi del programma è mostrato in Figura 3.12. In Figura 3.13 invece è possibile osservare quello della funzione PLT_Send_Packet modificata.

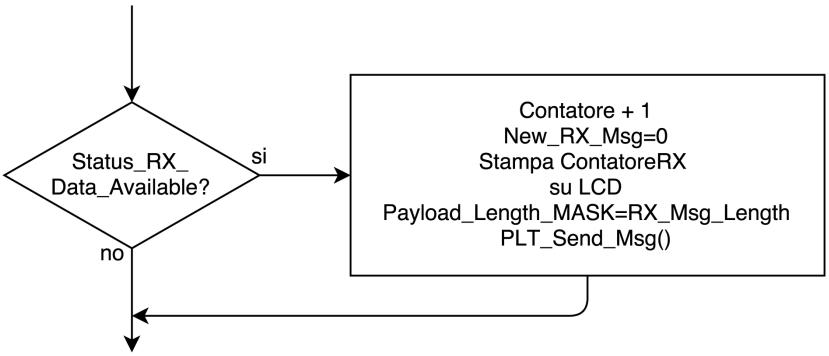


Figura 3.12: *Schema a blocchi della modifica al programma del chip per i test di polling*

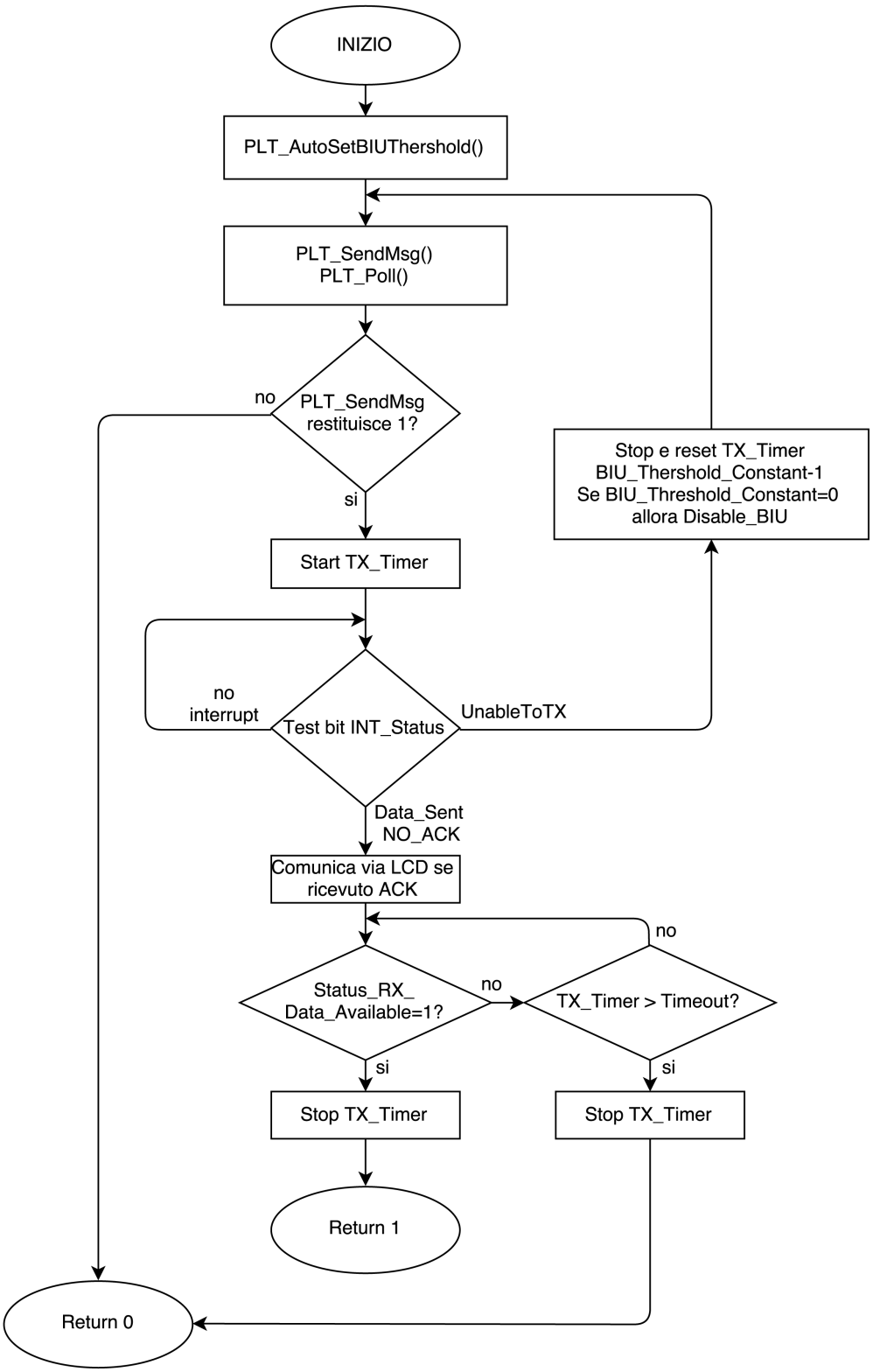


Figura 3.13: *Schema a blocchi della funzione* $PLT\_Transmit\_Packet()$ *modificata*

3.2 La rete utilizzata

Per le prove sperimentali si è disposto di una rete locale alimentata da un generatore UPS a 220V, accessibile in vari punti e inizialmente immune da disturbi. Mediante l'UPS si ottiene infatti una alimentazione molto "pulita", priva di rumore e disturbi, nella quale la comunicazione PLC avviene, come provato in seguito, senza alcun problema quindi con ottimi risultati. Per una analisi più realistica si è potuto aggiungere alla rete una fonte di rumore derivante da un inverter collegato a dei pannelli solari. Poiché questi forniscono una tensione continua in uscita, si rende necessario l'uso di un inverter che si occupa di trasformare il segnale in alternata; questo, a causa delle frequenti oscillazioni degli interruttori al suo interno, introduce rumore nella rete. Lo schema della rete di cui si è disposto è osservabile in Figura 3.14, nella quale i numeri indicano i nodi accessibili e la S l'UPS.

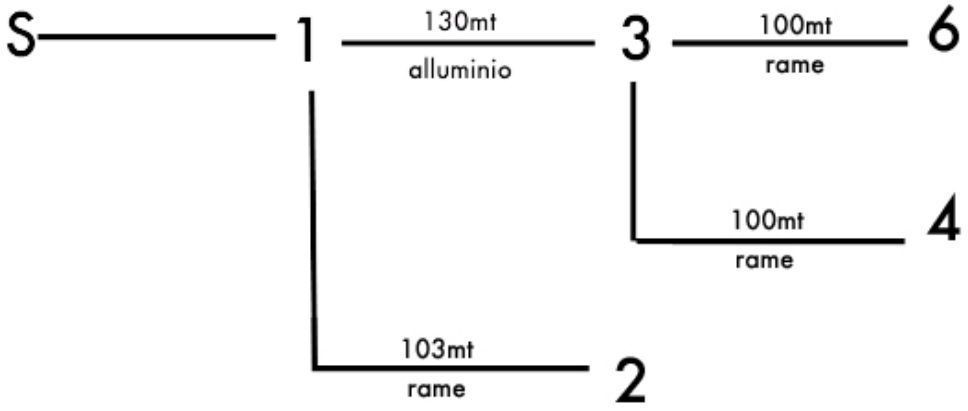


Figura 3.14: *Schema della rete utilizzata*

Ogni nodo è collegato ad una normale presa (CEE 7/16) alla quale può essere collegato uno o più dei modem Cypress. In questo modo si sono potute effettuare comunicazioni interponendo tra i due modem una lunghezza variabile di collegamento in rame o alluminio. Con la rete configurata in questo modo è possibile variare la lunghezza a salti di 100 mt circa, fino ad un massimo di circa 333 mt tra i due nodi più lontani. Le conseguenze derivanti dalla presenza dell'inverter vengono discusse nel capitolo 5 mentre nel prossimo vengono studiate le prestazioni dei modem in assenza di disturbi.
Lo schema del sistema complessivo utilizzato per i test è quindi quello illustrato in Figura 3.15.

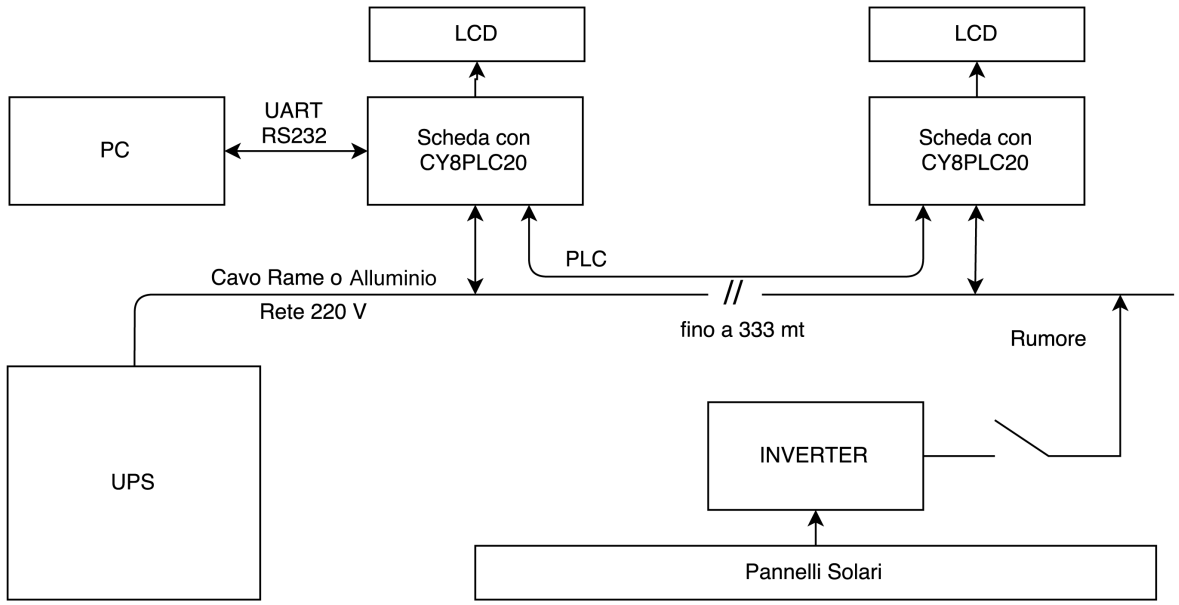


Figura 3.15: *Schema a blocchi dell'apparato complessivo utilizzato*

Capitolo 4

Prestazioni in assenza di disturbi

Segue l'analisi dei dati ricavati sperimentalmente. Tutte le misure complete rilevate sono osservabili in Appendice mentre vengono riportati in questo capitolo solamente i dati più importanti.

4.1 Senza Protocollo di Rete Cypress

Utilizzando il programma descritto nel paragrafo 3.1.1 è stato possibile analizzare la velocità di trasmissione dei bit senza utilizzare il protocollo di rete Cypress. In particolare, come detto in precedenza è possibile inviare di volta in volta un numero diverso di byte e di considerare il tempo necessario per inviarli. In questo modo sono stati inviate:

- 5 serie da 250 byte
- 5 serie da 200 byte
- 5 serie da 150 byte
- 5 serie da 100 byte
- 5 serie da 50 byte
- 5 serie da 10 byte
- 5 serie da 1 byte

In questa modalità d'uso la BIU detection e l'accesso CSMA al mezzo vengono realizzati dall'istruzione $PLT\_AcquirePowerline$. Nel programma utilizzato il timer viene avviato appena prima del comando $PLT\_SendData$ e fermato subito dopo. Sperimentalmente si è verificato che i tempi di trasmissione misurati con $BIU\_Interval\_Span = 85ms/BIU\_Interval\_Min = 30ms$ sono identici a quelli ottenuti con $BIU\_Interval\_Span = 15ms/BIU\_Interval\_Min = 5ms$ proprio perché l'istruzione $PLT\_AcquirePowerline$ non è compresa nelle misure, e che prove ripetute dello stesso invio portano a tempi che differiscono tra loro per meno di 1ms. Questo approccio permette di fatto l'esclusione del tempo aleatorio di accesso al mezzo pertanto

nelle seguenti formule verrà considerato il ritardo fisso dovuto solo al $TXDelay$. Nei calcoli vengono utilizzati i tempi minimi rilevati.

Secondo quanto ricavato dalla teoria i contributi parziali che compongono il tempo totale sono tre:

$$T_{teorico\_tot} = TXDelay + Nbytes \cdot 11 \cdot \frac{1}{Vbps} + Nbytes \cdot ByteRit \tag{4.1}$$

dove il primo contributo è dato dal $TXDelay$, il secondo dalla trasmissione di 11 bit per ognuno degli $Nbytes$ inviati alla velocità $Vbps$ ed il terzo dal ritardo tra ogni byte ($ByteRit$) del quale non è noto un valore preciso. A questo punto si possono scegliere due criteri di analisi:

1. Determinare l'effettiva $Vbps$ realizzata,
   assumendo il ritardo tra i byte inviati pari al $Tbit = \frac{1}{Vbps}$, poiché come descritto in precedenza, il costruttore dichiara che il tempo minimo tra la trasmissione di due byte consecutivi è di un Tbit

2. Determinare il ritardo tra i byte $ByteRit$,
   assumendo la $Vbps$ pari al valore nominale 2400 bps

Si dimostra che per entrambe le soluzioni risulta univocamente determinato il ritardo fisso. Inserendo infatti i risultati ottenuti su un grafico nel quale l'asse delle ascisse rappresenti il numero di byte inviati mentre quello delle ordinate rappresenti il tempo necessario per inviarli, interpolandoli si ottiene la retta la cui equazione ideale dovrebbe coincidere con la seguente, derivata dalla 4.1:

$$T_{tot} = (11 \cdot \frac{1}{Vbps} + ByteRit) \cdot Nbytes + TXDelay \tag{4.2}$$

$$Y = (\frac{11}{Vbps} + ByreRit) \cdot X + TXDelay \tag{4.3}$$

pertanto è possibile identificare i parametri della retta interpolatrice come:

$$m = \frac{11}{Vbps} + ByteRit \tag{4.4}$$

$$q = TXDelay \tag{4.5}$$

Si osserva come si renda necessario fissare uno dei parametri tra $Vbps$ e $ByteRit$ per poter determinare l'altro conoscendo il valore di m.
In Tabella 4.1 sono osservabili i tempi di invio minimi (che comprendono anche i ritardi fissi dovuti al $TXDelay$) ed i tempi di ricezione minimi (che invece non comprendono quel ritardo fisso). In Figura 4.1 si può osservare il grafico Tempo/Byte con le due curve: quella ottenuta dai tempi di invio e quella ottenuta dai tempi di ricezione; sono visibili nel grafico anche le relative equazioni delle rette interpolatrici $y = mx + q$.

$Nbytes$	$T_{invio}\,[ms]$	$T_{ricezione}\,[ms]$
250	1286,8125	1275,09375
200	1031,0625	1019,75
150	775,3125	764,0625
100	519,59375	508,28125
50	263,84375	252,5625
10	59,25	47,96875
1	13,21875	1,9375

Tabella 4.1: *Tempi di trasmissione e ricezione sperimentali, senza protocollo di rete*

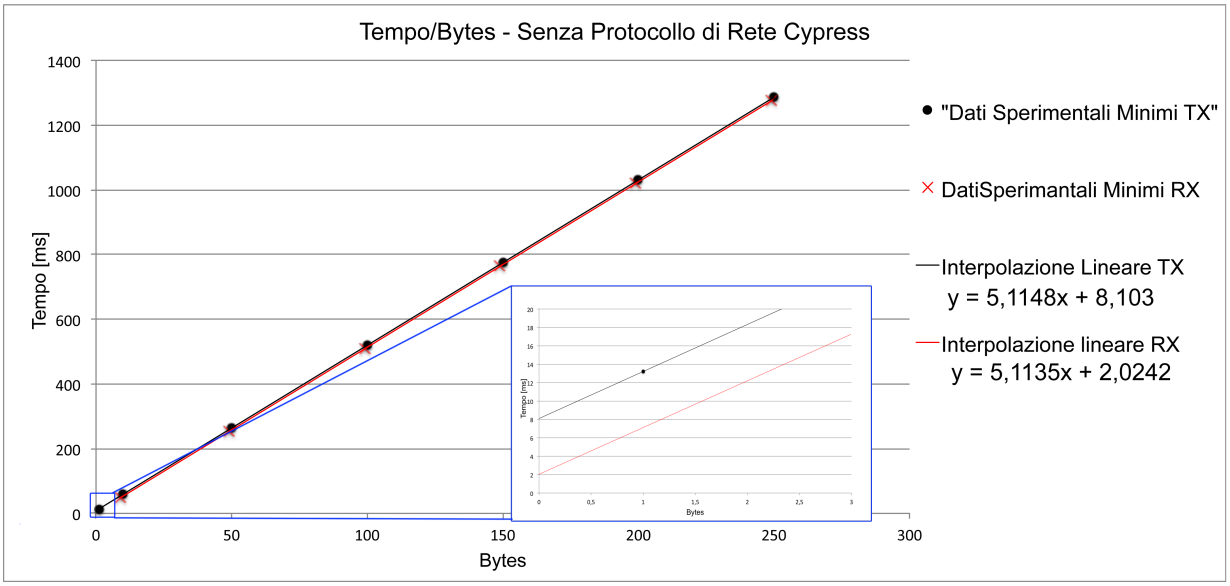


Figura 4.1: *Grafico Tempo/Bytes dati sperimentali di invio e ricezione, senza protocollo di rete, con interpolazione*

Da un punto di vista teorico le pendenze delle rette dovrebbero coincidere poiché sono determinate dalla velocità di trasmissione e dal ritardo fra i byte, non dai ritardi fissi che sono infatti l'unica differenza che si ha tra i tempi usati per la prima curva e quelli per la seconda. In effetti i coefficienti angolari ricavati sono molto simili e si è verificato che portano a risultati pressoché identici per quanto riguarda la determinazione di Vbps. Si fa notare inoltre che la retta dei tempi di ricezione presenta un termine noto non previsto dovuto probabilmente anche al fatto che il Timer viene stoppato dopo una breve sequenza di istruzioni relativa all'aggiornamento dei contatori e dell'LCD (si veda i codice in Appendice). Si prosegue quindi considerando la prima delle due rette interpolatrici, quella relativa ai tempi di trasmissione.

Dal grafico si può osservare quindi il valore di $TXDelay$ pari al termine noto della retta interpolatrice della prima curva:
$TXDelay = 8,103\ ms$.

Il valore previsto in realtà doveva essere di circa 7ms secondo quanto impostato mediante l'apposito registro: la differenza può essere attribuita ad una imprecisione nella generazione del $TXDelay$ sommata ad un ritardo dovuto ad una elaborazione della CPU. Questa teoria verrà confermata più avanti.
Nel seguito si propongono i risultati relativi alle due soluzioni di analisi.

1) Determinazione velocità effettiva
Per ottenere un indice qualitativo è innanzitutto possibile determinare una stima della velocità di trasmissione in Bps come:

$$VBps_{effettiva} = \frac{Numero\,Bytes\,Trasmessi}{Tempo\,Di\,Trasmissione\,Bytes}$$

che permette in secondo luogo di ottenere il tempo di trasmissione con la seguente formula semplificata:

$$T_{tot} = \frac{Numero\,Bytes\,Trasmessi}{VBps_{effettiva}} + TXDelay \tag{4.6}$$

con $Tempo\,Di\,Trasmissione\,Bytes = T_{tot} - TXDelay$

Si ottiene una $VBps_{effettiva}$ pari all'inverso della pendenza del grafico di Figura 4.1 ovvero:

$$VBps_{effettiva} = \frac{1\ Byte}{5,1148\ ms} \simeq 195,51\ Bps$$

Convertendo questo valore in bit per secondo considerando ogni Byte composto da 12 bit (1 start, 8 dati, 1 parità, 1 stop, 1 ritardo minimo) si ottiene:

$Vbps_{effettiva} = 195,51 \cdot 12 \simeq 2346,12\ bps$

Si fa notare che considerare la Vbps così ricavata coincide con l'imporre:

$$ByteRit = \frac{1}{Vbps}$$

nella 4.2 e ricavare Vbps, infatti si ha:

$$Ttot = (\frac{11}{Vbps} + ByteRit) \cdot Nbyte + TXDelay$$
$$Ttot = (\frac{11}{Vbps} + \frac{1}{Vbps}) \cdot Nbyte + TXDelay$$
$$Ttot = (\frac{12}{Vbps}) \cdot Nbyte + TXDelay$$

dalla quale si osserva che:

$$m = \frac{12}{Vbps}$$

e quindi:

$$Vbps = \frac{1}{m} \cdot 12$$

2) Determinazione ritardo tra i byte
Assumendo invece una velocità di trasmissione effettiva e pari a 2400 bps, è possibile stimare il valore del ritardo tra i byte. Per ottenere questo valore è possibile ricavare il valore di $ByteRit$ dalla 4.4 ottenendo:

$$ByteRit = m - \frac{11}{Vbps} = 0,53147\ ms \simeq 531,47\ \mu s \tag{4.7}$$

In alternativa, per ottenere una stima più precisa, è possibile considerare le differenze tra i tempi sperimentali riferiti ad un certo numero di byte e quelli teorici (considerando 11 bit ad ogni byte e nessun ritardo): plottando quindi il grafico Differenze/Byte, dall'interpolazione si ottiene una retta la cui pendenza determina il ritardo fisso ad ogni Byte infatti:

$$\begin{aligned} T_{teo} &= Nbytes \cdot \frac{11}{Vbps} \\ T_{sper} &= (\frac{11}{Vbps} + ByteRit) \cdot Nbytes + TXDelay \\ T_{sper} - T_{teo} &= ByteRit \cdot Nbytes + TXDelay \end{aligned}$$

pertanto in questo grafico si ha:

$$m = ByteRit$$

Il grafico descritto è osservabile in Figura 4.2, ed il risultato ottenuto porta ad un ritardo fisso tra i byte di:
$ByteRit = 531,5\ \mu s$,
molto simile al valore precedentemente ricavato.

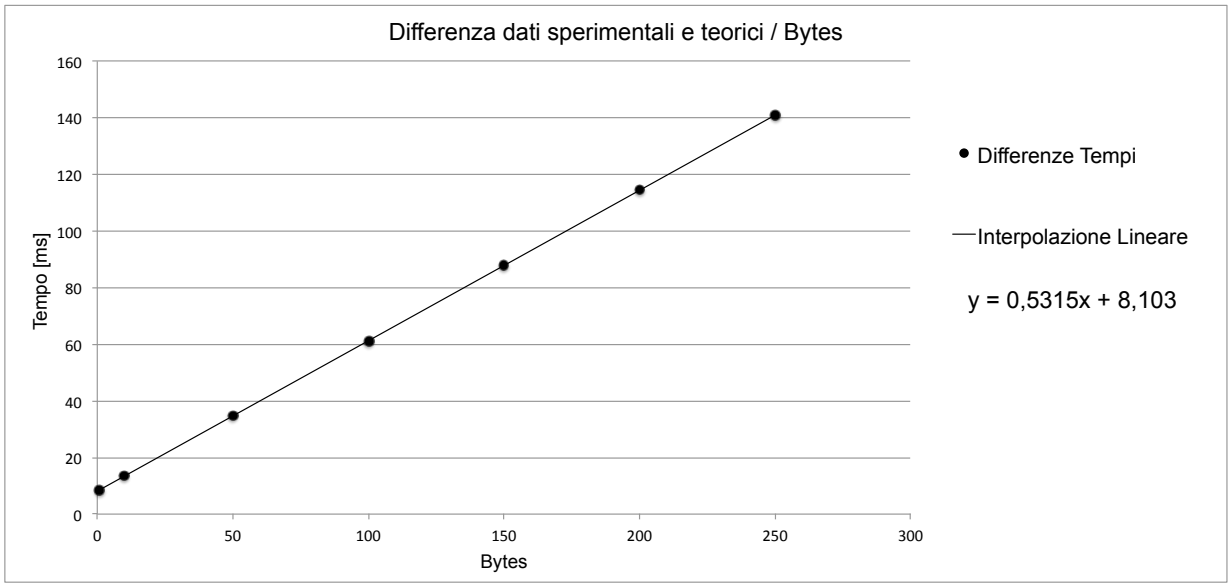


Figura 4.2: *Grafico Differenze/Bytes tra dati sperimentali e teorici, senza protocollo di rete, con interpolazione*

Come detto in precedenza, il ritardo tra un byte e l'altro deve essere almeno di un bit, infatti ad una velocità di 2400 bps si ha:

$$Tbit = \frac{1}{Vbps} \simeq 416,7\ \mu s$$

A questo punto è possibile assumere che i rimanenti $114,8\ \mu s$ ($531,5\mu s - 416,7\mu s$) siano un tempo di elaborazione tra un byte e l'altro, dipendente quindi dalla frequenza di lavoro della CPU.
Per avere conferma di questo si sono ripetute tutte le prove di variazione del numero di byte inviati e relative analisi dei grafici variando però anche il clock della CPU. I risultati ottenuti, mostrati in Tabella 4.2, non hanno confermato questa ipotesi.

f_{Clock}	$TXDelay\,[ms]$	$ByteRit\,[\mu s]$
24 MHz	8,103	531,5
24 MHz / 4	9,8976	531,4
24 MHZ / 8	12,152	531,4
24 MHz / 32	32,965	957,6

Tabella 4.2: *Ritardo tra byte e ritardo fisso totale al variare della frequenza della CPU*

Come si può osservare il valore del ritardo tra i byte sembra non dipendere dalla frequenza del clock, mentre il ritardo fisso di trasmissione di 8 ms aumenta al diminuire della frequenza del clock. Plottando in un grafico la relazione tra il ritardo e le frequenza della CPU si osserva una

proporzionalità inversa in cui compare però anche un offset. Per determinare l'offset è possibile plottare il ritardo stimato in relazione al divisore del clock utilizzato: la retta interpolatrice indica che degli 8,1 ms circa 7,54 sono un ritardo fisso mentre la parte rimanente è un ritardo dovuto ad una elaborazione che dipende quindi dalla frequenza della CPU (non è comunque possibile scendere sotto agli 8,1 ms poiché ottenuti alla massima frequenza di lavoro della CPU).

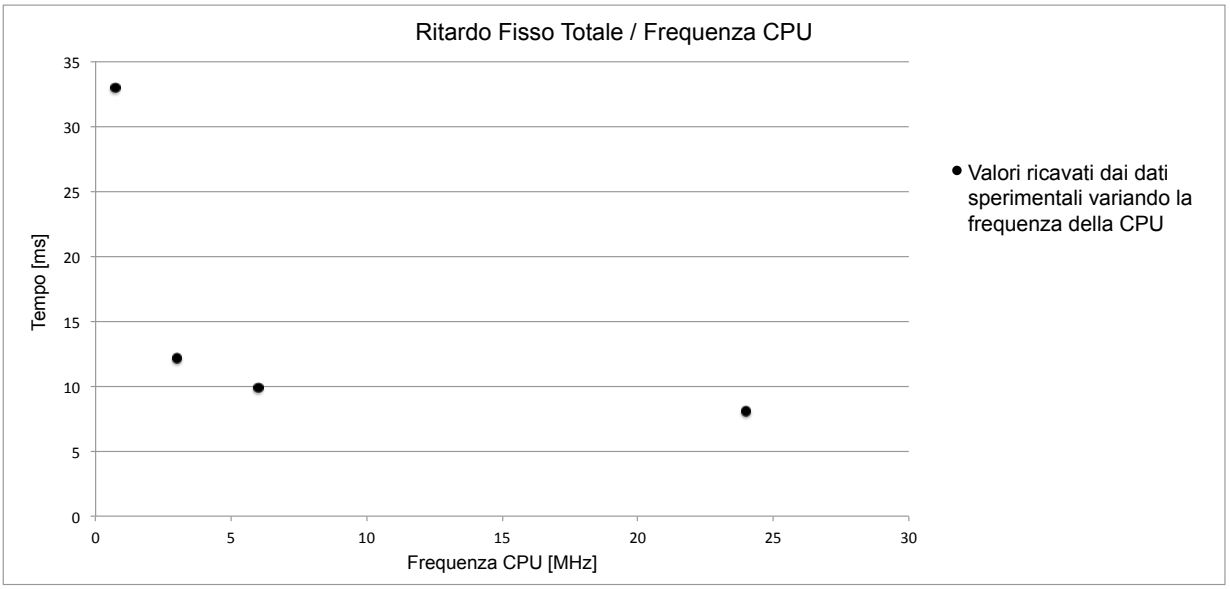


Figura 4.3: *Grafico Ritardo Fisso Totale / Frequenza CPU, si nota la proporzionalità inversa*

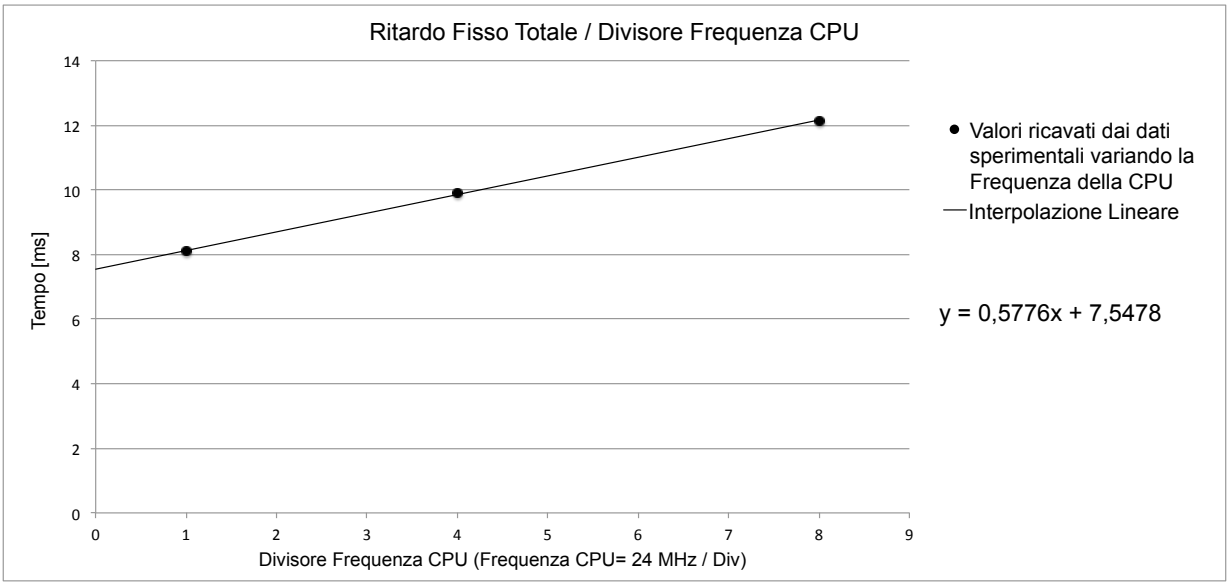


Figura 4.4: *Grafico Ritardo Fisso Totale / Divisore Frequenza CPU, proporzionalità lineare*

Questo risultato non è molto rilevante in quanto si da più importanza al ritardo tra i byte, che va a pesare maggiormente sulla velocità di trasmissione, piuttosto che al ritardo fisso. Si è quindi deciso di non approfondire l'analisi del ritardo fisso tra i byte, limitandosi ad averlo individuato e dimensionato.
Per quanto riguarda gli errori di trasmissione in assenza di rumore nella rete, si è potuto verificare che il 100% dei byte inviati arriva correttamente a destinazione, anche variando la disposizione dei modem nella rete.

Ricapitolando i risultati ottenuti sono mostrati in Tabella 4.3, 4.4 e 4.5.

$$\boxed{T_{tot} = (11 \cdot \frac{1}{Vbps} + ByteRit) \cdot Nbytes + TXDelay}$$

Tabella 4.3: *Formula completa Tempo di Invio di Nbytes senza protocollo di rete Cypress*

Parametro	**Valore Determinato**
$TXDelay$	$\sim 8,103\ ms$
Velocità effettiva in Bps	$\sim 195,5\ Bps$
Velocità effettiva in bps	$\sim 2346,13\ bps$
Ritardo fisso tra i byte, considerando velocità di 2400bps	$\sim 531,5\ \mu s$
Differenza tra ritardo fisso tra byte e ciclo di clock	$\sim 114,8\ \mu s$

Tabella 4.4: *Parametri determinati senza protocollo di rete*

$$\boxed{T_{tot} = 5,1148 \cdot Nbytes + 8,103,}$$

Tabella 4.5: *Formula diretta Tempo di Invio di Nbytes, [ms]*

4.2 Con Protocollo di Rete Cypress

Il programma utilizzato, in combinazione con l'applicazione Windows descritta nel capitolo precedente, permette di conoscere il tempo trascorso tra l'inizio dell'invio di un pacchetto e uno dei seguenti eventi:

1. Fine della trasmissione del pacchetto,
   $Service\_Type$=Senza Ack

2. Ricezione dell'Ack,
   $Service\_Type$=Con Ack e dispositivo ricevente acceso

3. Timeout di ricezione dell'ack,
   $Service\_Type$=Con Ack e dispositivo ricevente spento

Per ottenere risultati più rilevanti, questa fase, visto che utilizzando il protocollo di rete ciò è possibile, viene svolta trascurando l'accesso CSMA al mezzo ed escludendo quindi tutte le aleatorietà nella determinazione del tempo totale. Per fare questo si disabilita la BIU detection mediante l'apposito bit $Disable\_BIU$ del registro $PLC\_Mode\_Register$. L'analisi dei risultati ottenuti nei tre casi sopraccitati permette di dedurre conclusioni relative a vari parametri. Si precisa che anche in questo tipo di comunicazione si è verificato un successo del 100% delle trasmissioni con minime differenze nei tempi (meno di 1 ms), pertanto nelle analisi si è scelto di utilizzare i valori minimi ottenuti, osservabili di volta in volta nelle relative tabelle. Una panoramica più vasta di tutti i valori ottenuti viene presentata in Appendice. Nel seguito vengono analizzati i risultati ottenuti nelle tre modalità elencate sopra.

4.2.1 Tempo di invio di un pacchetto

Analogamente al paragrafo precedente, in questo vengono analizzati i tempi di trasmissione di un pacchetto Cypress. I tempi minimi ottenuti sono stati i seguenti:

$Nbytes\ del\ Payload$	$T_{sperimentali}\ [ms]$
0	51,062
7	87,031
15	128,156
23	169,25
31	210,375

Tabella 4.6: *Tempi di trasmissione minimi sperimentali, protocollo di rete Cypress, Ack Disattivato*

La struttura del pacchetto è quella descritta nel Capitolo 2, pertanto un invio comporta un tempo totale dato dall'Equazione 4.8, simile alla 4.1, dove si individuano i contributi dati dal

$TXDelay$, dall'invio di tutti i byte del pacchetto (tra cui 8 byte di controllo dovuti a header, preambolo e footer introdotti dal protocollo di rete) alla velocità Vbps e dal ritardo tra i bytes.

$$T_{tot} = TXDelay + (8 + Nbyte) \cdot 11 \cdot \frac{1}{Vbps} + ByteRit \cdot (8 + Nbyte) \tag{4.8}$$

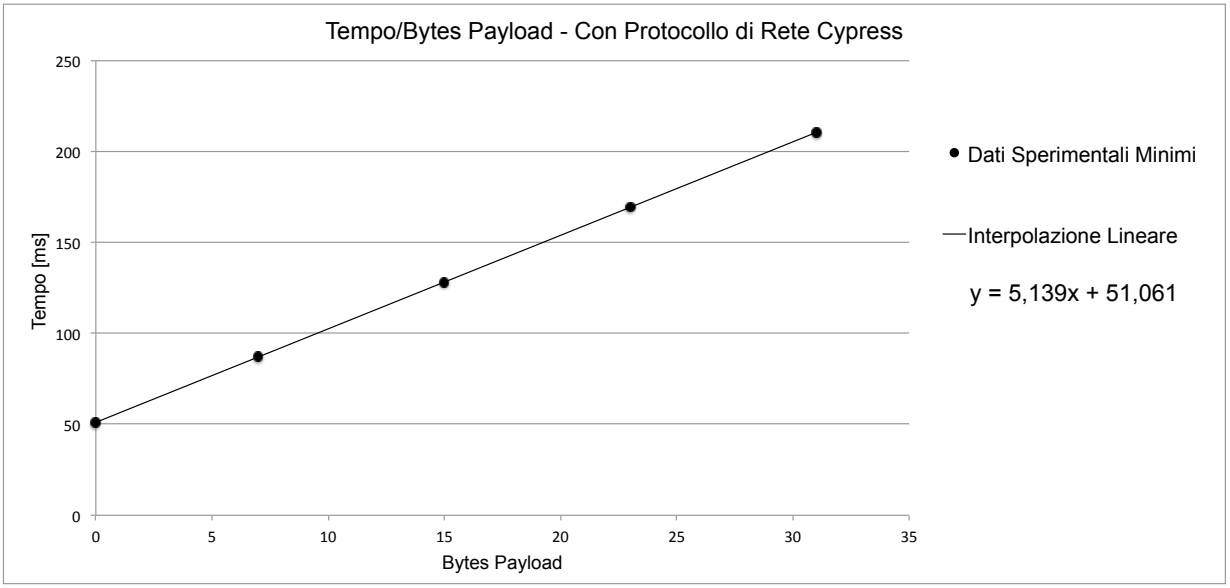


Figura 4.5: *Grafico Tempo/Bytes dati minimi sperimentali con protocollo di rete, con interpolazione*

Plottando sul grafico Tempo/Bytes i risultati ottenuti (Figura 4.5) è possibile determinare la retta interpolatrice di equazione $y = mx + q$, coincidente con l'equazione 4.9. ricavata dalla 4.8:

$$T_{tot} = (\frac{11}{Vbps} + ByteRit) \cdot Nbyte + [TXDelay + 8 \cdot (\frac{11}{Vbps} + ByteRit)] \tag{4.9}$$

$$Y = (\frac{11}{Vbps} + ByteRit) \cdot X + [TXDelay + 8 \cdot (\frac{11}{Vbps} + ByteRit)] \tag{4.10}$$

$$m = \frac{11}{Vbps} + ByteRit = 5,139\ ms/Byte \tag{4.11}$$

$$q = TXDelay + 8 \cdot (\frac{11}{Vbps} + ByteRit) = 51,061\ ms \tag{4.12}$$

A questo punto, come nel paragrafo precedente, dopo aver fissato uno dei parametri è possibile trovare l'altro; il valore definito $TXDelay$ in realtà può comprendere anche un ritardo di elaborazione fisso e la partizione tra questo ed il reale valore di $TXDelay$ non viene approfondita. Si procede quindi con la stessa analisi vista in precedenza.

1) Determinazione velocità effettiva
Assumendo un ritardo tra i byte di $ByteRit = Tbit = \frac{1}{Vbps}$, si ha che Vbps è data ancora dall'inverso della pendenza del grafico di Figura 4.5 portato in bps da Bps (con Byte=12 bit, come visto nel paragrafo precedente). Si ha infatti ancora:

$$T_{tot} = (\frac{11}{Vbps} + ByteRit) \cdot Nbyte + [TXDelay + 8 \cdot (\frac{11}{Vbps} + ByteRit)]$$
$$T_{tot} = \frac{12}{Vbps} \cdot Nbyte + [TXDelay + 8 \cdot \frac{12}{Vbps}] \quad (4.13)$$

Si ottengono:

$$VBps_{effettiva} = \frac{1}{m} = \frac{1\,Byte}{5,139\,ms} \simeq 194,6\ Bps$$
$$Vbps_{effettiva} = 194,6 \cdot 12 \simeq 2335,1\ bps$$

Il valore di $TXDelay$ invece è ricavato dalla 4.13:

$$TXDelay = q - 8 \cdot \frac{12}{Vbps} = 51,061 - 8 \cdot \frac{12}{2335,1} \simeq 9,95\ ms \quad (4.14)$$

Che è maggiore del valore ricavato senza il protocollo di rete.

2) Determinazione ritardo tra i byte
Come nel paragrafo precedente è possibile procedere altrimenti fissando la velocità di trasmissione $Vbps$ pari a 2400 e ricavando il valore sperimentale del ritardo tra i byte. Si possono ricavare i valori di $ByteRit$ direttamente dalla 4.9:

$$ByteRit = (m - \frac{11}{Vbps}) \simeq 555,67\ \mu s \quad (4.15)$$

$$TXDelay = q - 8 \cdot (\frac{11}{Vbps} + ByteRit) = 9,95\ ms \quad (4.16)$$

In alternativa, come già visto, si può ottenere una stima più precisa confrontando i dati teorici (con $ByteRit$ e $TXDelay$ entrambi nulli) con quelli sperimentali punto per punto ed interpolando i valori Differenza/Byte. Ne risulta il grafico di figura:

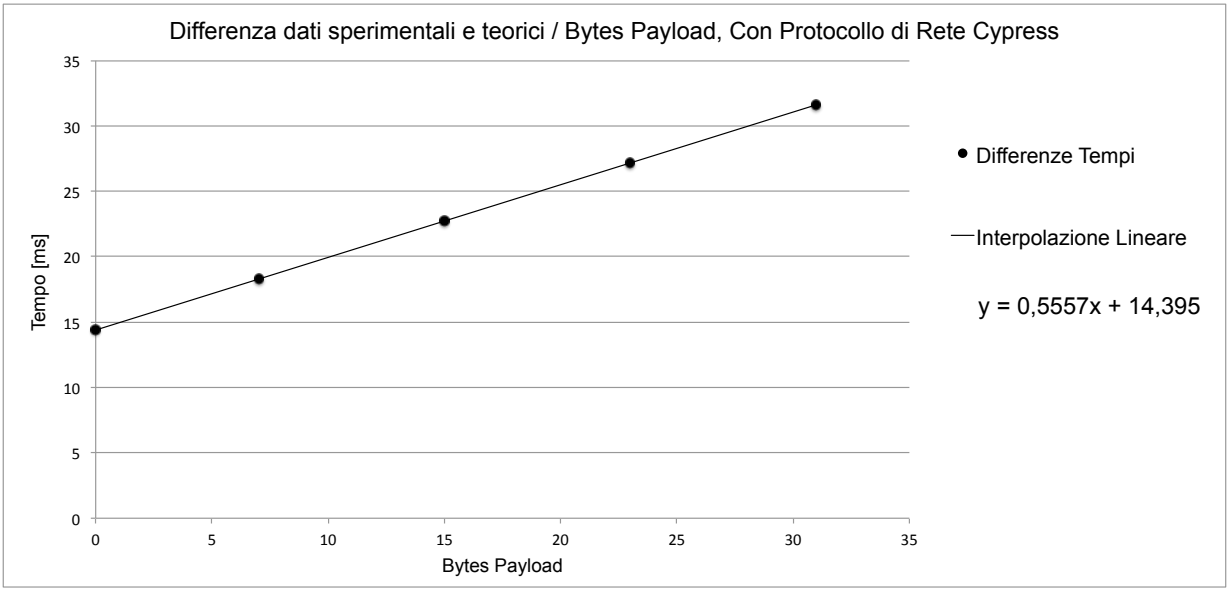


Figura 4.6: *Grafico Differenze/Bytes tra dati minimi sperimentali e teorici, con protocollo di rete, con interpolazione*

dove valgono:

$$T_{teo} = \frac{11}{Vbps} \cdot Nbyte + 8 \cdot \frac{11}{Vbps}$$

$$T_{sper} = (\frac{11}{Vbps} + ByteRit) \cdot Nbyte + [TXDelay + 8 \cdot (\frac{11}{Vbps} + ByteRit)]$$

$$T_{sper} - T_{teo} = ByteRit \cdot Nbytes + [TXDelay + 8 \cdot ByteRit] \quad (4.17)$$

$$m = ByteRit = 555,7\ \mu s \quad (4.18)$$

$$q = TXDelay + 8 \cdot ByteRit = 14,395\ ms \quad (4.19)$$

che portano a:

$$TXDelay = q - 8 \cdot ByteRit = 9,95\ ms$$

I metodi utilizzati determinano quindi i seguenti risultati:

$$\boxed{T_{tot} = (\frac{11}{Vbps} + ByteRit) \cdot Nbyte + [TXDelay + 8 \cdot (\frac{11}{Vbps} + ByteRit)]}$$

Tabella 4.7: *Formula completa Tempo di Invio Pacchetto Nbytes Payload*

Parametro	Valore Determinato
TXDelay + Ritardo di elaborazione	$\sim 9,95\ ms$
Velocità effettiva in Bps	$\sim 194,6\ Bps$
Velocità effettiva in bps	$\sim 2335,1\ bps$
Ritardo fisso tra i byte, considerando velocità di 2400bps	$\sim 555,67\ \mu s$
Differenza tra ritardo fisso tra byte e ciclo di clock	$\sim 139,00\ \mu s$
Differenza tra il ritardo fisso tra byte con e senza protocollo di rete	$\sim 24,17\ \mu s$

Tabella 4.8: *Parametri determinati con il protocollo di rete*

$$\boxed{T_{tot} = 5,1139 \cdot Nbytes + 51,061}$$

Tabella 4.9: *Formula diretta Tempo di Invio Pacchetto Nbytes Payload, [ms]*

Come confermato successivamente dai tecnici Cypress, il protocollo di rete introduce un'elaborazione tra l'invio di un byte e l'altro che causa un ulteriore ritardo. L'aumento del $TXDelay$ invece potrebbe comprendere un ritardo di elaborazione del pacchetto che richiede, tra le altre cose, anche la generazione dei bit di controllo errori CRC.

4.2.2 Tempo di invio di un pacchetto e ritorno dell'ack

Si ripete la trattazione effettuata negli altri casi ricordando che l'Ack consiste in un pacchetto con payload di 0 bytes. I risultati sperimentali migliori sono osservabili in Tabella 4.10, seguono le equazioni specifiche del caso, i grafici Tempo/Bytes e DifferenzaSperimentaleDaTeorico/Bytes con relative rette interpolatrici e risultati ottenuti dalle analisi 1) e 2) viste in precedenza.
Si osserva inoltre che anche in questo caso il 100% delle trasmissioni va sempre a buon fine, non si è mai avuto alcun fallimento in più di 20000 trasmissioni registrate.

$Nbytes\ del\ Payload$	$T_{sperimentali}\ [ms]$
0	99,094
7	135,219
15	176,469
23	217,75
31	259

Tabella 4.10: *Tempi di trasmissione sperimentali, protocollo di rete Cypress, Ack Ricevuto*

Equazioni dei tempi:

$$T_{tot} = 2 \cdot TXDelay + (2 \cdot 8 + Nbyte) \cdot 11 \cdot \frac{1}{Vbps} + ByteRit \cdot (2 \cdot 8 + Nbyte) \qquad (4.20)$$

che può essere riscritta come:

$$T_{tot} = (\frac{11}{Vbps} + ByteRit) \cdot Nbytes + [2 \cdot TXDelay + 16 \cdot (ByteRit + \frac{11}{Vbps})] \quad (4.21)$$

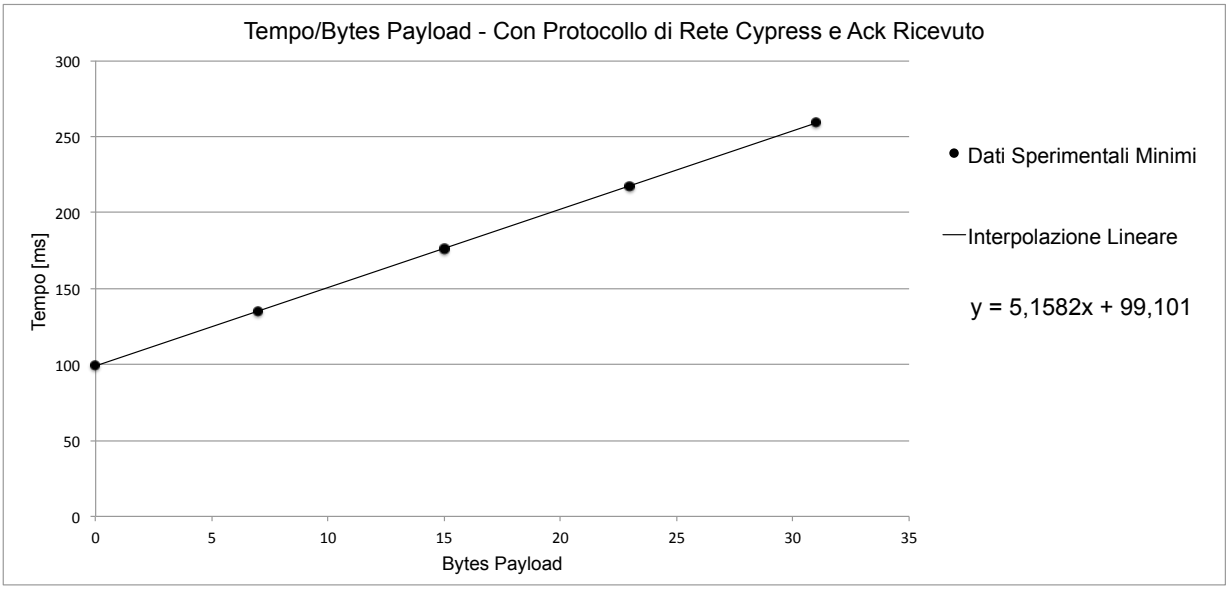


Figura 4.7: *Grafico Tempo/Bytes dati minimi sperimentali con protocollo di rete e ricezione ack, con interpolazione*

Equazioni della retta interpolatrice del grafico Tempo/Bytes $y = mx + q$:

$$m = (\frac{11}{Vbps} + ByteRit) = 5,1582 \ ms/byte \quad (4.22)$$

$$q = [2 \cdot TXDelay + 16 \cdot (ByteRit + \frac{11}{Vbps})] = 99,101 \ ms \quad (4.23)$$

$$(4.24)$$

1) Determinazione velocità effettiva
Assumendo un ritardo tra i byte pari ad un $Tbit$ si ha come sempre:

$$T_{tot} = (\frac{12}{Vbps}) \cdot Nbyte + [2 \cdot TXDelay + 16 \cdot (\frac{12}{Vbps})] \tag{4.25}$$

$$m = \frac{12}{Vbps} \tag{4.26}$$

$$q = 2 \cdot TXDelay + 16 \cdot (\frac{12}{Vbps}) \tag{4.27}$$

$$VBps_{effettiva} = \frac{1}{m} = \frac{1\,Byte}{5,1582\,ms} \simeq 193,87\ Bps \tag{4.28}$$

$$Vbps_{effettiva} = 194,6 \cdot 12 \simeq 2326,4\ bps \tag{4.29}$$

$$\tag{4.30}$$

Il valore di $TXDelay$ invece è ricavato dalla 4.22:

$$TXDelay = [q - 16 \cdot (\frac{12}{Vbps})]/2 = [99,101 - 16 \cdot (\frac{12}{2326,4})]/2 = 8,285\ ms \tag{4.31}$$

Che è ancora in disaccordo con il valore ricavato senza il protocollo di rete, tuttavia è ammissibile come detto in precedenza che questo valore incapsuli un ulteriore ritardo fisso di elaborazione.

2) Determinazione ritardo tra i byte
Come nei paragrafi precedenti è possibile procedere altrimenti fissando la velocità di trasmissione $Vbps$ pari a 2400 e ricavando il valore sperimentale del ritardo tra i byte. Si possono ricavare i valori di $ByteRit$ direttamente dalla 4.22:

$$ByteRit = (m - \frac{11}{Vbps}) \simeq 574,86\ \mu s \tag{4.32}$$

$$TXDelay = [q - 16 \cdot (\frac{11}{Vbps} + ByteRit)]/2 \simeq 8,285\ ms \tag{4.33}$$

$$\tag{4.34}$$

oppure, come già visto, ottenere una stima più precisa confrontando i dati teorici (con $ByteRit$ e $TXDelay$ nulli) con quelli sperimentali punto per punto ed interpolando i valori Differenza/Byte. Ne risulta il grafico di figura:

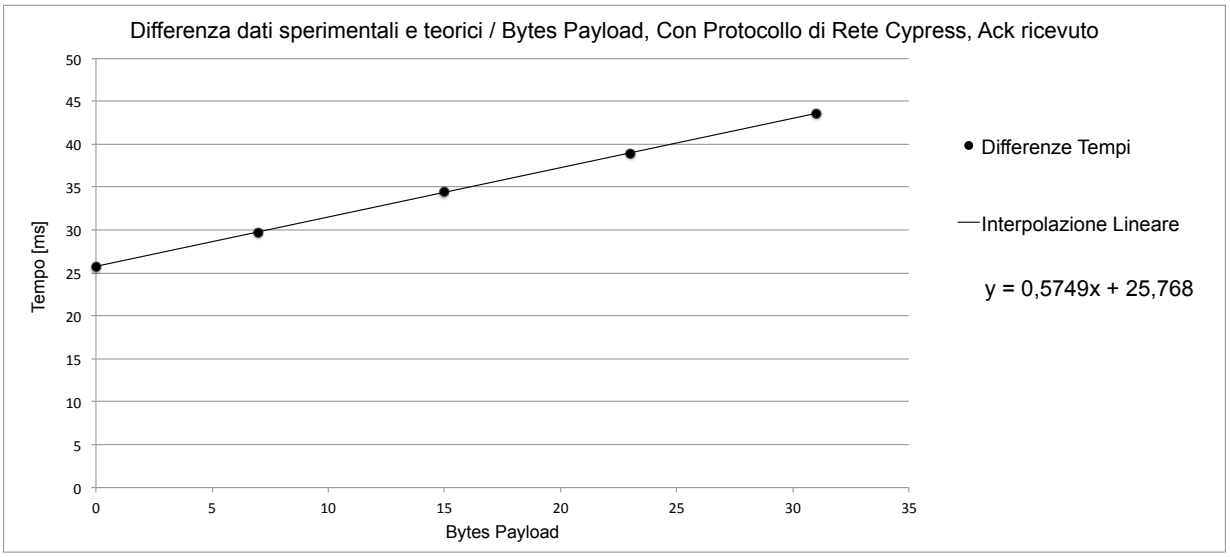


Figura 4.8: *Grafico Differenze/Bytes tra dati minimi sperimentali e teorici, con protocollo di rete e ricezione ack, con interpolazione*

dove:

$$Tteo = \frac{11}{Vbps} \cdot Nbyte + 16 \cdot \frac{11}{Vbps}$$

$$T_{sper} = (\frac{11}{Vbps} + ByteRit) \cdot Nbyte + [2 \cdot TXDelay + 16 \cdot (\frac{11}{Vbps} + ByteRit)]$$

$$T_{sper} - T_{teo} = ByteRit \cdot Nbytes + [2 \cdot TXDelay + 16 \cdot ByteRit] \quad (4.35)$$

$$m = ByteRit \quad (4.36)$$

$$q = 2 \cdot TXDelay + 16 \cdot ByteRit = 25,768ms \quad (4.37)$$

che portano a:

$$ByteRit = 574,9\ \mu s$$

$$TXDelay = (q - 16 \cdot ByteRit)/2 = 8,285\ ms$$

$$\boxed{T_{tot} = (\frac{11}{Vbps} + ByteRit) \cdot Nbytes + [2 \cdot TXDelay + 16 \cdot (ByteRit + \frac{11}{Vbps})]}$$

Tabella 4.11: *Formula completa Tempo di Invio Pacchetto Nbytes Payload e ritorno Ack*

Parametro	Valore Determinato
TXDelay + Ritardo di elaborazione	$\sim 8,285\ ms$
Velocità effettiva in Bps	$\sim 193,87\ Bps$
Velocità effettiva in bps	$\sim 2326,4\ bps$
Ritardo fisso tra i byte, considerando velocità di 2400bps	$\sim 574,9\ \mu s$
Differenza tra ritardo fisso tra byte e ciclo di clock	$\sim 158,23\ \mu s$
Differenza tra il ritardo fisso tra byte con e senza protocollo di rete	$43,4\ \mu s$

Tabella 4.12: *Parametri determinati con il protocollo di rete e ricezione dell'ack*

$$\boxed{T_{tot} = 5,1582 \cdot Nbytes + 99,101}$$

Tabella 4.13: *Formula diretta Tempo di Invio Pacchetto Nbytes Payload e ritorno Ack, [ms]*

Come si può notare compaiono delle piccole differenze tra i ritardi fissi nel servizio con Ack da quello senza; questo è molto strano in quanto l'aggiunta dell'Ack di ritorno quasi sicuramente non influisce con il sistema di trasmissione. Una giustificazione può derivare dal fatto che, nella trattazione di questo paragrafo, in particolare nella formula 4.20, si è supposto che la trasmissione del pacchetto e dell'Ack avvengano esattamente allo stesso modo quando invece potrebbero avvenire in modo diverso da un punto di vista delle operazioni software eseguite a basso livello. In alternativa si sarebbe potuto assumere invariato il metodo di invio del pacchetto e sottrarre ai tempi totali di questo paragrafo (pacchetto + ack) la porzione relativa al pacchetto (dalle misure effettuate, non mediante calcolo teorico) e studiarne la porzione rimanente. Con questo criterio si avrebbero le seguenti misure di partenza:

$Nbytes\ del\ Payload$	$T_{ack}\ [ms]$
0	48,032
7	48,188
15	48,313
23	48,5
31	48,625

Tabella 4.14: *Tempi di trasmissione solo ack sperimentali*

$$Tack = Tdelay + 8 \cdot (\frac{11}{Vbps} + ByteRit)$$

Come si può notare in questa struttura del tempo di ack non vi è dipendenza dal parametro Nbyte, mentre vi è un'aumento del tempo sperimentale all'aumentare del numero di byte di payload del pacchetto inviato. Si può assumere questa differenza come tempo di elaborazione, dovuto ad esempio alla verifica dei bit di controllo errori CRC che potrebbe richiedere un tempo

proporzionale alla dimensione del pacchetto. Si arriva così ad un'equazione del tipo:

$$Tack = Tdelay + 8 \cdot (\frac{11}{Vbps} + ByteRitAck) + Nbytes \cdot CostanteElaborazione \quad (4.38)$$

$$Tack = Tack_{fisso} + Nbytes \cdot CostanteElaborazione \quad (4.39)$$

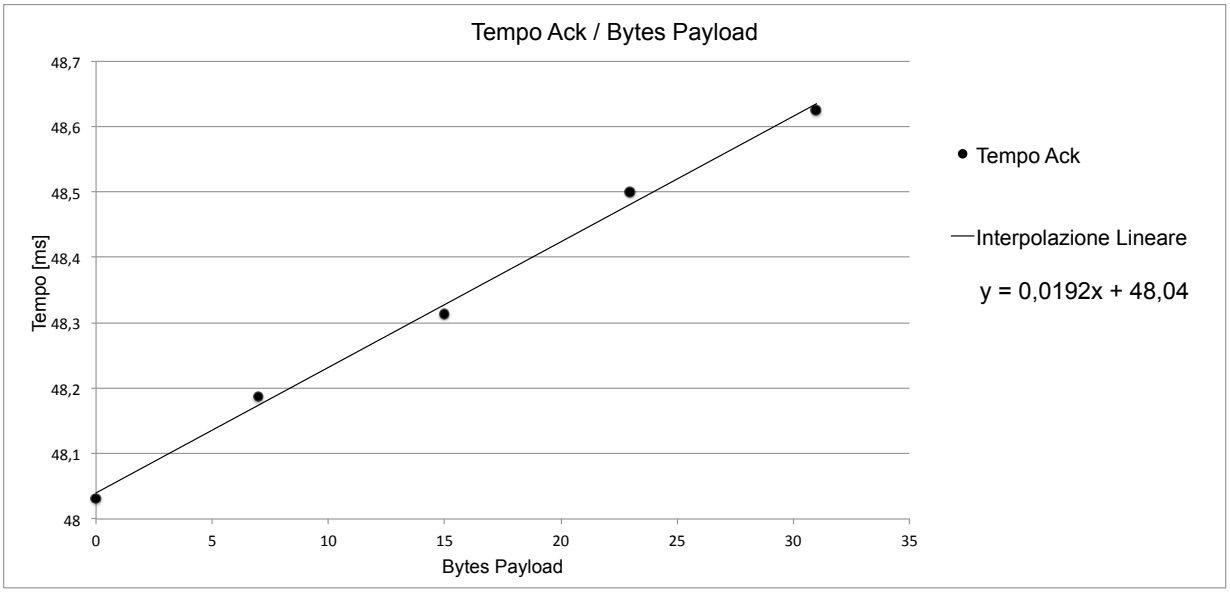


Figura 4.9: *Grafico Tempo di Ack / Bytes , con protocollo di rete e ricezione ack, con interpolazione*

Plottando quindi i dati su un grafico Tack/Bytes (Figura 4.9) è possibile interpolare i dati e ricavare la parte fissa del tempo di ack e quella variabile con la lunghezza del pacchetto ricevuto, ottenendo:

$$\begin{aligned} Tack_{fisso} &= 48,04\ ms \\ CostanteElaborazione &= 19,2\ \mu s/Byte\ di\ payload. \end{aligned} \quad (4.40)$$

La parte fissa è in accordo ad esempio con un $TXDelay$ di 8 ms ed un ritardi fisso tra un byte e l'altro pari a $Tbit = 1/2400$, infatti:

$$Tack_{min} = 8 + 8 \cdot 11/2400 \simeq 48\ ms$$

Con questo approccio si è evidenziato che il ritardo nella trasmissione dell'ack dipende dal numero di byte ricevuti. Si è scelto di non approfondire ulteriormente lo studio di questi risultati in quanto dipendono strettamente dalla realizzazione Cypress e non da limiti fisici della trasmissione PLC.

$$\boxed{Tack = Tack_{fisso} + Nbytes \cdot CostanteElaborazione}$$

Tabella 4.15: *Formula diretta Tempo di Invio Ack per Pacchetto Nbytes Payload, valgono ancora le formule di Tabella 4.7 e 4.8.*

Parametro	Valore Determinato
$Tack_{fisso}$	$\sim 48,04\ ms$
$CostanteElaborazione$	$\sim 19,2\ \mu s/Byte$

Tabella 4.16: *Parametri determinati con il protocollo di rete e ricezione dell'ack*

$$\boxed{T_{ack} = 0,0192 \cdot Nbytes + 48,04}$$

Tabella 4.17: *Formula diretta Tempo di Invio Ack per Pacchetto Nbytes Payload, [ms]*

4.2.3 Tempo di invio di un pacchetto e timeout di ricezione ack

Per quanto riguarda questa modalità, la variazione del parametro $Ack\_Timeout$ del registro $Timing\_Config$ permette la valutazione del timeout impostabile. Ricordando che sono selezionabili le seguenti 4 voci:

- $Auto + 20ms$
- $Auto + 50ms$
- $Auto + 100ms$
- $500ms$

Il costruttore dichiara che nel calcolo del tempo $Auto$ il chip tiene conto della lunghezza del pacchetto inviato, della baudrate selezionata e del valore di $TXDelay$ in uso. Inizialmente si pensava di poter sfruttare questo parametro per confermare quanto stabilito fino ad ora riguardo i tempi di trasmissione confrontandolo con le formule ricavate in precedenza. Variando la lunghezza del pacchetto inviato in realtà si è scoperto che $Auto$ non ha una "variazione analogica" ma può assumere uno tra 4 valori discreti a step di 2 ms quando ad esempio si fissa $TXDelay$ pari a 7 ed una baudrate di 2400bps. Il tempo misurato dal Timer comprende l'invio del pacchetto e l'attesa dell'ack che si suppone iniziare al termine dell'invio del pacchetto stesso.
Per misurare il tempo di generazione dell'evento Timeout di ricezione Ack, il dispositivo ricevente è stato spento ed il numero di TX_Retry è impostato a zero, in modo che non si abbia il ritorno dell'ack ma uno ed un solo ciclo di attesa. Si è verificato inoltre che variando solo il valore di $Ack\_Timeout$ tra $Auto + 20ms$, $Auto + 50ms$ ed $Auto + 100ms$, il tempo di attesa varia rispettivamente di 18,50 e 100 ms da un valore centrale (nei rilevamenti eseguiti per le

successive considerazioni si è impostato $Auto + 100ms$).
Si ha pertanto:

$$T_{tot} = T_{pacchetto} + T_{waitack} = T_{pacchetto} + [T_{auto} + 18/50/100ms] \tag{4.41}$$

$$T_{auto} = T_{tot} - T_{pacchetto} - 100ms \tag{4.42}$$

Si è deciso di procedere valutando ogni possibile payload (non solo 0, 7, 15, 23 e 31); dopo aver calcolato tutti i $T_{pacchetto}$ utilizzando i risultati ottenuti nei paragrafi precedenti (formula 4.9), si è ricavato il valore di T_{auto} per ogni misura e si è potuto realizzare il grafico di Figura 4.10 che mostra quanto accennato riguardo la variazione non analogica di questo tempo di attesa automatico.

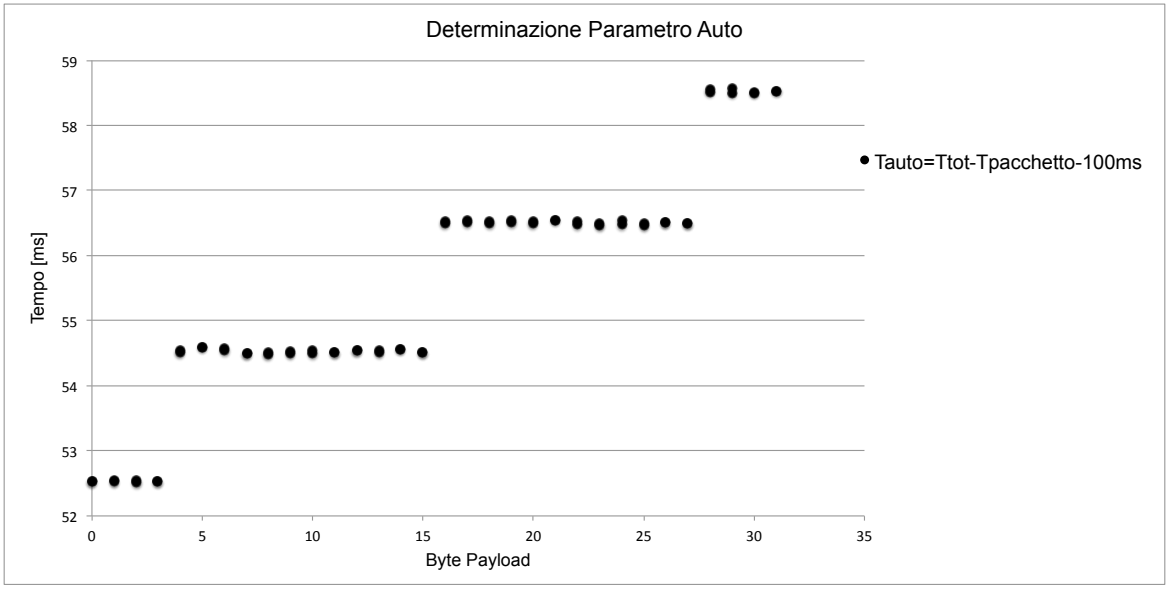


Figura 4.10: *Grafico valutazione parametro Auto / Bytes*

4.2.4 Percentuali di fallimento ed indipendenza dei tempi dalla distanza

Le prove sui tempi della trasmissione pacchetto-ack sono state eseguite a varia distanza (100, 200, 300 mt circa) e, oltre a non aver registrato alcun fallimento di trasmissione ne di ricezione dell'ack, non si è nemmeno registrata alcuna differenza importante nei tempi di invio. I valori minimi, medi e massimi al variare della distanza tra i due nodi della rete sono osservabili in Appendice, Tabella C.5.
Si ricorda che Cypress dichiara la piena affidabilità della trasmissione mediante questi dispositivi testandola su una rete non disturbata e con una distanza massima di circa 3,5 km. Nelle prove eseguite, tali prestazioni sono state effettivamente ottenute fino ad una distanza di 330 metri. Distanze maggiori non sono state oggetto i test per motivi pratici.

4.2.5 Diagramma calcolo tempo di trasmissione

Per quanto studiato in questa sezione è possibile calcolare il tempo di trasmissione di un pacchetto in base alle varie impostazioni mediante il seguente diagramma a blocchi, dove si notano anche le conseguenze dell'impostazione di un valore di TX_Retry diverso da zero.

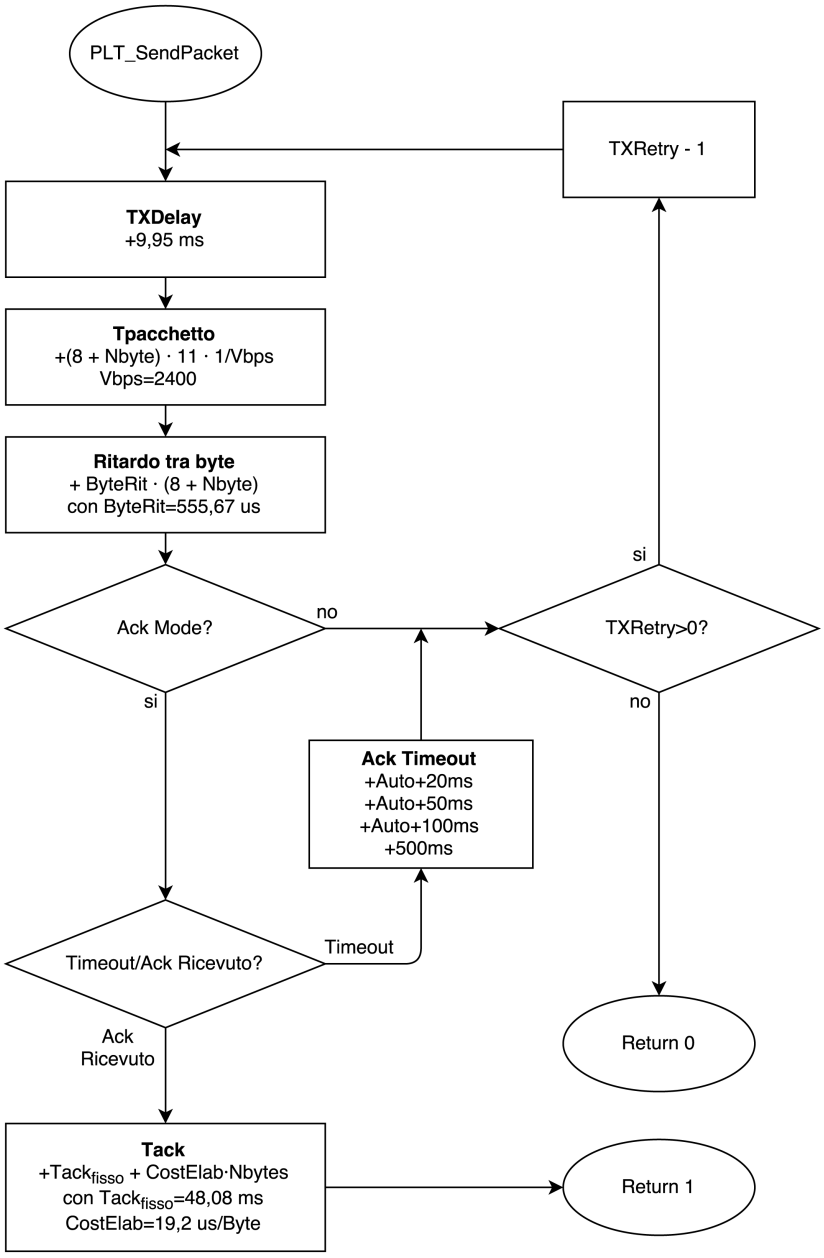


Figura 4.11: *Diagramma per il calcolo del tempo totale con protocollo di rete*

4.3 Test di polling

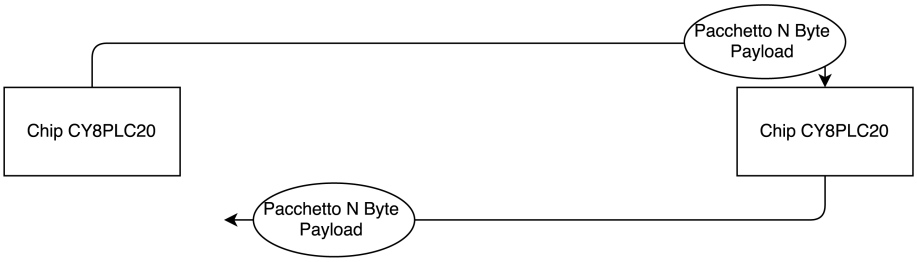


Figura 4.12: *Schema comunicazione di tipo polling*

Mediante il programma descritto nel Capitolo precedente si è proceduto con il test di polling. L'analisi dei tempi in questo caso si è ritenuta superflua in quanto per ottenere il tempo totale di polling basta sommare i vari contributi dovuti a pacchetti ed ack di cui si conoscono già i valori pratici effettivi (teoria verificabile rapidamente). In questi test si nota una minima frequenza di fallimento, anche nell'invio di pacchetti di risposta di 0 byte che di fatto dovrebbero coincidere con la trasmissione in Acknowledgement Mode utilizzata in precedenza: ciò conferma il sospetto che l'invio dell'ack e quello dei pacchetti normali avvengano in modo leggermente diverso e quest'ultimo, in un test continuo di 5000 comunicazioni, comporta una piccola quantità di errori dovuta in generale all'aumento della complessità della comunicazione. Si sono eseguite varie prove cambiando la posizione dei modem nella rete ed i risultati principali ottenuti sono osservabili in Appendice in Tabella C.2 e riassunti in Tabella 4.18, dove compaiono ordinate le prove per tipologia e:
"Da" indica la posizione del dispositivo trasmettitore, in accordo con lo schema della rete di Figura 3.14
"A" indica la posizione del dispositivo ricevitore
"Inverter" indica la posizione dell'inverter
"Ack" indica il numero di ack presenti nella trasmissione
"RXGain" indica la soglia di ricezione
"TXGain" indica il guadagno di trasmissione
"BIU" indica se è attiva la BIU detestino, quindi l'accesso CSMA al mezzo.
"Retry" indica il valore di retry
"TXDelay" indica la durata del TXDelay impostata
"Timestamp" indica la data e l'ora della prova
"%fall" indica la percentuale di fallimenti delle trasmissioni
"Successi" indica il numero di trasmissioni complete rispetto al numero di trasmissioni totali
"RX" indica il numero di pacchetti correttamente ricevuti dal ricevitore
"PER" indica il $Packet\ Error\ Rate$ relativo ai pacchetti di sola andata

Da	A	Ack	RXGain	TXGain [V]	BIU	Retry	TXDelay [ms]	Timestamp	% fall.	Successi	RX	PER
1	1	No	250 uV	1.55	Off	0	7	2015/09/28 13:20:17	0.94	4953/5000	4963	0.74
1	1	No	1.25V	1.55	Off	0	7	2015/09/28 13:51:36	0.82	4959/5000	4976	0.48
1	2	No	250 uV	1.55	On	0	7	2015/09/28 15:39:07	0.26	4987/5000	4998	0.04
1	2	No	250 uV	1.55	Off	0	7	2015/09/28 10:50:01	0.58	4971/5000	4982	0.36
1	2	No	250 uV	1.55	Off	0	7	2015/07/22 17:16:13	1.08	4946/5000	4971	0.58
1	2	No	250 uV	1.55	Off	0	7	2015/09/16 10:06:36	0.76	4962/5000	4977	0.46
1	2	No	250 uV	1.55	Off	0	25	2015/09/28 16:09:41	0.58	4971/5000	4977	0.46
1	2	2	250 uV	1.55	Off	0	7	2015/07/15 16:54:27	2.1	4895/5000	4915	1.7
1	2	2	250 uV	1.55	Off	0	7	2015/07/2216:49:57	0.78	4961/5000	4976	0.48
1	2	No	1.25 V	1.55	Off	0	7	2015/09/28 11:19:03	1.12	4944/5000	4963	0.74
1	2	No	1.25 V	1.55	Off	0	7	2015/09/2814:22:16	1.16	4942/5000	4966	0.68
1	2	No	1.25 V	3.50	Off	0	7	2015/09/28 14:52:51	0.72	4964/5000	4974	0.52
1	3	No	1.25 V	1.55	Off	0	7	2015/09/28 11:48:11	1.08	4946/5000	4966	0.68
1	6	No	250 uV	1.55	Off	0	7	2015/09/28 12:51:17	1.32	4934/5000	4952	0.96
1	6	2	250 uV	1.55	Off	0	7	2015/07/28 10:21:25	0.96	4952/5000	4982	0.36
1	6	No	1.25 V	1.55	Off	0	7	2015/09/28 12:18:56	0.72	4964/5000	4977	0.72
2	3	No	250 uV	1.55	Off	0	7	2015/09/16 10:33:27	0.9	4955/5000	4975	0.5
2	6	No	250 uV	1.55	Off	0	7	2015/09/16 11:01:08	0.7	4965/5000	4983	0.34
2	6	No	250 uV	1.55	Off	0	25	2015/09/16 12:32:52	1.26	4937/5000	4959	0.82
2	6	2	250 uV	1.55	Off	5	7	2015/09/16 13:13:56	0	5000/5000	5000	0
2	6	2	250 uV	1.55	On	5	7	2015/09/16 14:05:00	0.2	4990/5000	5000	0
2	6	No	5 mV	3.5	Off	0	7	2015/09/16 12:01:35	1.58	4921/5000	4942	1.16

Tabella 4.18: *Percentuali di fallimento test di polling, senza inverter nella rete*

La penultima colonna contiene il numero di pacchetti ricevuti correttamente dal ricevitore pertanto permette di ricavare il reale $Packet\ Error\ Rate$ della comunicazione, indicato nell'ultima colonna. $PER = \frac{TotTx - TotRx}{TotTx}$ Come si può osservare dai dati in Appendice (e nella seguente Tabella 4.19) si ottengono percentuali di fallimento maggiori all'aumentare del numero di byte del payload, in particolare si va da percentuali dell'ordine dell'1% per pacchetti di 0 byte a percentuali del 2/3% per i pacchetti con payload di 31 byte. Si è verificato che le percentuali di fallimento non dipendono dalla distanza tra i due nodi poiché ci sono casi in cui si hanno errori maggiori con distanze più brevi. Anche la distribuzione nel tempo di questi fallimenti (dei singoli pacchetti in una serie da 5000) è casuale, non si hanno casi di fallimenti strettamente consecutivi o altamente concentrati, motivo in più per giustificare la difficile identificazione della causa di questo fenomeno.
Si è poi scelto di osservare i risultati degli stessi test dopo aver variato le soglie di $TXGain$ ed $RXGain$, consentendo una comunicazione teoricamente più solida e meno influenzabile dal rumore; i risultati ottenuti non mostrano invece alcun miglioramento anzi, in linea di massima con la soglia $RXGain$ a 5 si osserva sempre un peggioramento. Notando inoltre che ad ogni ripetizione della stessa prova i risultati in termini di percentuali cambiano, si è proseguito cercando la causa degli errori altrove. Supponendo che i fallimenti fossero allora dovuti ad un problema nella velocità con cui i dispositivi inviano e ricevono dati, alternando il funzionamento tra ricezione e trasmissione, si sono eseguite delle prove aumentando al massimo il $TXDelay$ (25 ms) che rallenta molto la comunicazione. Anche in questa prova non si è avuto alcun tipo di miglioramento, anzi; diventa quindi molto difficile determinare la causa di questi fallimenti.
Con lo scopo di evidenziare l'utilità di un efficace protocollo di rete, si è effettuata anche una prova impostando il Tx_Retry a 5, aumentando al massimo la soglia di $RXGain$ ed utilizzando la modalità con Ack (in modo da riprovare l'invio di ogni pacchetto fino 5 volte prima di registrare il fallimento). Si è così ottenuto il risultato migliore, ovvero 0 fallimenti su 5000 invii. Nonostante l'ottimo indice di successo, si fa notare che i tempi medi in questo caso si sono alzati di molto (es. 31 byte, tempo medio da 422ms a 527ms, 25% maggiore) in quanto si sono effettuati spesso delle ritrasmissioni. Se inoltre si aggiunge l'accesso CSMA al mezzo, sempre con un TX_Retry pari a 5, si ottengono percentuali di successo del 99,8% e tempi medi di circa

726 ms (massimi addirittura dell'ordine del secondo), che potrebbero anche essere un problema per applicazioni di domotica (rispettanti la normativa CENELEC) le quali richiedono una certa velocità di risposta, come ad esempio una regolazione della luminosità di una stanza (ricordando nuovamente che in questo caso la rete è priva di disturbi e che nei tempi citati è compreso il ritorno dell'ack). I risultati principali in termini di tempi di trasmissione, sui quali si basano le precedenti considerazioni, sono osservabili in Appendice C.6 e sono in parte mostrati nella seguente Tabella 4.19.

Byte Payload	**Esito**	**%**	**Tmin [ms]**	**Tmax [ms]**	**Tmed [ms]**
Da 1 a 2, No Ack, BIUOff, TXDelay=7ms, 2015/07/22 17:16:13					
0	Successo	99.6	101.938	102.5	102.2246
0	Fallimento	0.4	1000.031	1000.125	1000.0833
7	Successo	99.4	174.188	174.812	174.4602
7	Fallimento	0.6	1000.062	1000.219	1000.1012
15	Successo	99.2	256.719	257.312	256.9891
15	Fallimento	0.8	1000.062	1000.219	1000.1211
23	Successo	98.6	339.188	339.844	339.5638
23	Fallimento	1.4	1000.031	1000.219	1000.1339
31	Successo	97.8	421.75	422.375	422.0847
31	Fallimento	2.2	1000.031	1000.219	1000.1178
Da 2 a 6, 2 ack, 5 retry, BIU Off, TXDelay=7ms, 2015/09/16 13:13:56					
0	Successo	100	203.906	325.594	204.6264
0	Fallimento	0	/	/	/
7	Successo	100	276.062	435.688	277.9167
7	Fallimento	0	/	/	/
15	Successo	100	358.719	559.156	361.7854
15	Fallimento	0	/	/	/
23	Successo	100	441.219	684.406	443.6705
23	Fallimento	0	/	/	/
31	Successo	100	523.688	809.906	527.1396
31	Fallimento	0	/	/	/
Da 2 a 6, 2 ack, 5 retry, BIU On, TXDelay=7ms, 2015/09/16 14:05:00					
0	Successo	100	384.344	639.844	409.6609
0	Fallimento	0	/	/	/
7	Successo	100	460.594	766.125	483.8395
7	Fallimento	0	/	/	/
15	Successo	100	539.219	877.719	566.2877
15	Fallimento	0	/	/	/
23	Successo	100	621.75	1003.188	650.0105
23	Fallimento	0	/	/	/
31	Successo	99	329.969	1089.219	726.2774
31	Fallimento	1	1000.062	1000.219	1000.1626

Tabella 4.19: *Analisi dei principali Log di Trasmissione senza inverter nella rete, polling*

Capitolo 5

Prestazioni con disturbo nella rete

5.1 Senza Protocollo di Rete Cypress

Per quanto riguarda l'applicazione senza protocollo di rete si è osservato che, una volta abilitata la ricezione di qualsiasi byte dalla rete (escludendo il controllo di match 0x55 o 0xCC), con $RXGain$ a 1,25 mVrms ed in presenza di rumore, vengono ricevuti molti falsi byte (0xFF o simili) con frequenza casuale (anche più di 50 falsi byte in un minuto); si tratta chiaramente di un indice di come l'inverter nella rete possa effettivamente disturbare le comunicazioni. Per ridurre questo problema, nelle condizioni di sperimentazione del presente lavoro di tesi, si è alzata la soglia di $RXGain$ a 5mVrms e si è osservato che in effetti la frequenza di falsi byte ricevuti diminuisce; diversamente, impostando questa soglia a 250 μVrms, vengono ricevute quantità elevatissime di falsi byte, anche circa 8000 al minuto.
Per le prove il controllo di match è stato abilitato e si è scelto di memorizzare sia il numero di byte che superano questo controllo (RX corretti) sia il numero totale di byte ricevuti (che in presenza di rumore è generalmente maggiore del contatore precedente per quanto detto sopra). Per quanto riguarda le statistiche di fallimento si è osservato che tutti i byte inviati vengono ricevuti correttamente, il problema sta proprio nella ricezione dei falsi byte che si osserva quando il dispositivo rimane in ricezione senza che l'altro trasmettitore invii byte, anche se per intervalli brevissimi. Questo fenomeno andrebbe a danneggiare eventuali pacchetti nel caso si utilizzasse un protocollo di rete che li prevedesse, pertanto una analisi più approfondita delle percentuali di successo e fallimento nell'invio dei pacchetti viene eseguita nei prossimi paragrafi utilizzando il protocollo di rete Cypress.
L'ultima nota riguarda la frequenza dei falsi byte che come detto, dipende dalla soglia di $RXGain$ utilizzata e dal rumore nella rete. Variando questa soglia si sono misurati il numero di falsi byte ricevuti in un minuto posizionando il ricevitore in vari punti della rete per ottenere una stima dell'intensità del disturbo. Purtroppo si è notato che le stesse identiche prove ripetute in momenti diversi, anche consecutivi, portano a risultati completamente discordanti: si passa da qualche decina di falsi byte al minuto in alcune prove a qualche centinaia in altre. Gli studi non sono stati approfonditi in quanto il rumore non era né controllabile né descrivibile. Tuttavia si è appurato che aumentando la soglia di $RXGain$ si riduce il fenomeno di ricezione dei falsi byte.

5.2 Con Protocollo di Rete Cypress

I risultati completi di queste prove sono osservabili in Appendice in Tabella C.3 e consecutive mentre di seguito è osservabile una Tabella di riepilogo. Nella penultima colonna si ha ancora il numero di pacchetti correttamente ricevuti che permette di ricavare il $Packet\ Error\ Rate$ mostrato nell'ultima colonna.

Da	A	Inverter	RXGain	TXGain [V]	BIU	Retry	TXDelay [ms]	Timestamp	% fall.	Successi	RX	PER
1	2	6	250 uV	1.55 V	Off	0	7	2015/07/09 16-21-35	0	5000/5000	5000	0
1	3	6	250 uV	1.55 V	Off	0	7	2015/07/09 16:41:45	0	5000/5000	5000	0
1	4	6	250 uV	1.55 V	Off	0	7	2015/07/10 17:26:12	0	5000/5000	5000	0
1	6	6	250 uV	1.55 V	Off	0	7	2015/07/09 16:02:39	22.66	3867/5000	4072	18.56
1	6	6	250 uV	1.55 V	Off	0	7	2015/07/10 12:05:04	3.8	4810/5000	4823	3.53
1	6	6	250 uV	1.55 V	Off	0	7	2015/07/28 11:33:05	11.1	4445/5000	4492	10.16
1	6	6	250 uV	1.55 V	Off	0	7	2015/07/28 12:21:21	19	4050/5000	4189	16.22
1	6	6	250 uV	1.55 V	Off	0	7	2015/07/28 14:39:01	25.2	3740/5000	3979	20.42
1	6	6	250 uV	1.55 V	On	3	7	2015/07/09 17:43:56	0.1	4995/5000	4998	0.04
1	6	6	250 uV	1.55 V	Off	3	7	2015/07/09 15:05:31	0.42	4979/5000	5113	/
1	6	6	1.25 V	1.55 V	Off	0	7	2015/09/17 18:18:25	9.06	4547/5000	4568	8.64
1	6	6	1.25 V	1.55 V	Off	0	7	2015/09/17 16:28:10	11.88	4406/5000	4437	11.26
1	6	6	1.25 V	1.55 V	Off	5	7	2015/09/17 17:08:56	0	5000/5000	5000	0
1	6	6	1.25 V	1.55 V	On	5	7	2015/09/17 17:42:30	0	5000/5000	5000	0
6	1	1	250 uV	1.55 V	Off	0	7	2015/07/29 11:15:16	0	5000/5000	5000	0
6	1	6	1.25 V	1.55 V	Off	0	7	2015/09/17 16:46:58	12.9	4355/5000	4918	1.64

Tabella 5.1: *Percentuali di fallimento pacchetto-ack, con inverter nella rete*

Inizialmente si osserva come la percentuale di $Packet\ Error\ Rate$ del pacchetto di andata (Calcolata dal numero di ricezioni contate dal ricevitore) si discosti a volte di molto ed altre meno dalla percentuale di fallimenti della comunicazione totale. Si è potuto osservare che quando entrambi i dispositivi si trovano in punti lontani dalla fonte del disturbo, si hanno percentuali di errore molto ridotte per quanto riguarda l'invio di un pacchetto e la corretta ricezione dell'ack. Si può osservare ad esempio che le prime due prove della tabella non hanno riportato alcun errore. Provando a variare la disposizione dei dispositivi e della fonte si hanno avuto risultati spesso inaspettati; realizzando topologie pressoché identiche da un punto di vista delle distanze tra i dispositivi ma disposte in maniera diversa nella rete, si ottengono risultati completamente opposti. Ad esempio le disposizioni $16i6$ (TX in 1, RX in 6, inverter in 6) e $41i1$ (TX in 4, RX in 1, inverter in 1) consistono entrambe nell'avere il ricevitore posto vicino alla fonte di rumore ed il trasmettitore a 200 metri (di cavi per metà in alluminio e metà in rame) ma le percentuali di errore sono rispettivamente, in molteplici prove effettuate, dell'ordine del 30% e dello 0%. Inizialmente si era pensato ad un cattivo collegamento nel tragitto 16, in particolare nel passaggio dal tratto 13 al tratto 36, poiché la configurazione $36i6$ riportava errori inferiori al 1% mentre la $26i6$ era molto simile a quella di partenza. Anche questa tesi però è stata smentita dalle prove con la configurazione $61i1$, specchio di quella più problematica, che non ha riportato alcun errore, come se la rete fosse pulita. In generale si può osservare che il problema maggiore lo si ha nel tratto da 1 a 6 con l'inverter in posizione 6, infatti quando si testa la comunicazione da 6 a 1 con inverter in 6 si ha una grossa differenza tra i fallimenti in un verso e quelli nell'altro, evidenziando risultati peggiori nella direzione citata.
Si è appurato inoltre che la stessa prova di 5000 trasmissioni riporta risultati molto discordanti dai precedenti se ripetuta in momenti diversi della giornata o addirittura, a volte, anche con prove consecutive. Ad esempio con la configurazione $16i6$, la peggiore da un punto di vista dell'esito delle trasmissioni, si va da percentuali generali del PER del 3,2% ad altre del 25% (Nelle

percentuali relative ai soli pacchetti da 31 byte di payload addirittura si passa dall'8% al 40%). Per quanto riguarda questo fenomeno si può supporre una variazione del disturbo al variare ad esempio della temperatura dei pannelli o della loro disposizione rispetto al sole. Disturbo che però influenza in maniera diversa la comunicazione se iniettato in un determinato nodo piuttosto che in un altro.
L'unica "regolarità" si ha ancora nell'aumento della percentuale di fallimenti all'aumentare del numero di byte del payload. Anche in questo caso si sono ripetute le prove variando i valori di $TXGain$ ed $RXGain$, ottenendo comunque dei leggeri miglioramenti per $RXGain = 1,25mVrms$ che nel caso $16i6$ sembra permettere di rimanere sotto percentuali di errore del 10% contro quelle a volte superiori del 20% .
In Tabella 5.2 sono osservabili alcuni dei risultati principali in termini di tempi di trasmissione, commentati in seguito. Per una panoramico più completa si veda in Appendice, Tabella C.7.

Byte Payload	Esito	%	Tmin [ms]	Tmax [ms]	Tmed [ms]
Da 1 a 6, inverter in 6, BIU Off, TXDelay=7 ms, 2015/07/09 16:02:39					
0	Successo	86.8	99.094	99.406	99.2696
0	Fallimento	13.2	121.562	121.656	121.6073
7	Successo	81.7	135.188	135.562	135.3797
7	Fallimento	18.3	159.531	159.625	159.572
15	Successo	81.3	176.469	176.781	176.6484
15	Fallimento	18.7	200.625	200.719	200.6668
23	Successo	77	217.812	218.094	217.9483
23	Fallimento	23	243.719	243.875	243.787
31	Successo	59.9	259.062	266.5	259.2039
31	Fallimento	40.1	286.781	286.969	286.8872
Da 1 a 6, inverter in 6, 5 retry, BIU Off, TXDelay=7 ms, 2015/09/17 17:08:56					
0	Successo	100	99.094	463.031	115.742
0	Fallimento	0	/	/	/
7	Successo	100	135.188	612.5	160.4856
7	Fallimento	0	/	/	/
15	Successo	100	176.5	776.531	200.6559
15	Fallimento	0	/	/	/
23	Successo	100	217.812	1189.344	252.1712
23	Fallimento	0	/	/	/
31	Successo	100	259.031	1402.312	323.486
31	Fallimento	0	/	/	/
Da 1 a 6, inverter in 6, 5 retry, BIU On, TXDelay=7 ms, 20150917174230					
0	Successo	100	189.344	879.812	231.7505
0	Fallimento	0	/	/	/
7	Successo	100	225.375	1009.312	279.5632
7	Fallimento	0	/	/	/
15	Successo	100	266.719	1271.406	343.6341
15	Fallimento	0	/	/	/
23	Successo	100	1.625	1665.438	403.0429
23	Fallimento	0	/	/	/
31	Successo	100	349.188	2039.469	457.2952
31	Fallimento	0	/	/	/

Tabella 5.2: *Analisi dei principali Log di Trasmissione con inverter nella rete, pacchetto-ack*

Per ridurre infine la percentuale di errore al di sotto dello 0,2% anche nella configurazione più problematica si sono effettuate prove impostando il valore di $TXRetry$ a 3 e a 5, ottenendo tempi medi di successo per 31 byte di payload rispettivamente di 272 e 324 ms contro i 259 ms senza i retry (massimi però anche rispettivamente di 1,2/1,4 secondi). Quando si aggiunge anche

l'accesso CSMA si osservano, per payload di 31 byte, tempi medi di 457 ms e massimi di 2 secondi (contro i soliti 259 ms ottenuti a rete pulita senza CSMA). Valgono ancora e verranno riprese nel capitolo successivo le considerazioni fatte in precedenza riguardo i problemi derivanti da tempi di comunicazione così lunghi in applicazioni domotiche.

5.3 Test di polling

I risultati, osservabili in Appendice in Tabella C.8 e riepilogati nella seguente Tabella, sono leggermente peggiori del caso precedente.

Da	A	Inverter	Ack	RXGain	TXGain [V]	BIU	Retry	TXDelay [ms]	Timestamp	% fall.	Successi	RX	PER
1	2	6	No	250 uV	1.55 V	Off	0	7	2015/07/28 17/02/14	0.9	4955/5000	4976	0.48
1	6	6	2	250 uV	1.55 V	Off	5	7	2015/07/28 13:29:08	21.56	3922/5000	4298	14.04
1	6	6	No	1.25 V	1.55 V	Off	5	7	2015/09/17 13:36:49	4.18	4791/5000	4809	3.82
1	6	6	2	1.25 V	1.55 V	Off	5	7	2015/09/17 15:40:14	4.32	4784/5000	4957	0.86
1	6	6	No	5 mV	1.55 V	Off	0	7	2015/09/17 14:55:37	43.36	2832/5000	2901	41.98
2	1	1	No	250 uV	1.55 V	Off	0	7	2015/07/29 10:07:45	1.08	4946/5000	4974	0.52
2	6	6	2	250 uV	1.55 V	Off	0	7	2015/07/28 16:31:43	21.44	3928/5000	4269	14.62
3	6	6	No	1.25 V	1.55 V	Off	0	7	2015/09/17 13:08:39	0.3	4985/5000	4993	0.14
3	6	6	No	250 uV	1.55 V	Off	0	7	2015/07/28 17:29:43	4.9	4755/5000	4958	0.84
6	1	1	No	1.25 V	1.55 V	Off	0	7	2015/09/17 14:03:37	0.68	4966/5000	4966	0.68
6	6	6	No	1.25 V	1.55 V	Off	0	7	2015/09/17 12:43:44	1.02	4949/5000	4968	0.64

Tabella 5.3: *Percentuali di fallimento test di polling, con inverter nella rete*

Si notano la modesta variabilità delle percentuali relative alle stesse prove in momenti diversi e l'elevata differenza nella ripetizione della prova a specchio. Anche in questo caso l'unica regolarità si ha nella quantità di fallimenti maggiore all'aumentare del payload che, ricordiamo, è uguale per il pacchetto di andata e quello di ritorno. Analizzando il caso $16i6$ che come sempre riporta risultati peggiori, si si può notare che aumentando la soglia di $RXGain$ a 5 mV si ottiene una percentuale di errore del 43% contro il 4% della prova con $RXGain$ di 1,25 mVrms e 20% della prova a 250 μVrms. Ciò evidenzia il fatto che una soglia maggiore non implica una minore sensibilità ai disturbi come ci si poteva aspettare, i motivi sono tuttavia sconosciuti.
Per quanto riguarda i tempi si osservano i risultati principali in Appendice C.8, mostrati in parte anche nella seguente Tabella.

Byte Payload	Esito	%	Tmin [ms]	Tmax [ms]	Tmed [ms]
Da 1 a 6, inverter in 6, No Ack, BIU Off, TXDelay=7 ms, 2015/09/17 13:36:49					
0	Successo	98.3	102	102.5	102.2362
0	Fallimento	1.7	1000.031	1000.219	1000.1139
7	Successo	98.3	174.188	174.719	174.4578
7	Fallimento	1.7	1000.031	1000.219	1000.1342
15	Successo	96	256.688	257.312	257.0208
15	Fallimento	4	1000.031	1000.219	1000.1352
23	Successo	94	339.281	339.844	339.5616
23	Fallimento	6	1000.031	1000.219	1000.1261
31	Successo	92.5	416.938	422.406	422.1153
31	Fallimento	7.5	1000.031	1000.219	1000.1183
Da 2 a 6, inverter in 6, 2 Ack, BIU Off, TXDelay=7 ms, 2015/07/28 16:31:43					
0	Successo	86.6	203.906	204.5	204.182
0	Fallimento	13.4	1000.031	1000.25	1000.1182
7	Successo	85.3	276.094	276.719	276.3908
7	Fallimento	14.7	1000.031	1000.281	1000.1331
15	Successo	75.6	358.719	359.281	359.016
15	Fallimento	24.4	1000.031	1000.25	1000.1356
23	Successo	73.2	436.5	441.844	441.5373
23	Fallimento	26.8	1000.031	1000.281	1000.1331
31	Successo	72.1	523.719	524.406	524.0674
31	Fallimento	27.9	1000.031	1000.281	1000.1346

Tabella 5.4: *Analisi dei principali Log di Trasmissione con inverter nella rete, polling*

Utilizzando $TXRetry = 5$ è stato possibile portare la percentuale di errore solamente al 4%. La valutazione dei tempi in questo caso non è affidabile in quanto il timer a 16 bit impostato in questo modo permetteva misure massime di 2 secondi ed inoltre impostare un retry a 5 sia per il pacchetto di andata che per quello di ritorno comportava tempi troppo grandi per utilizzare un timeout anche di 2 secondi.
In linea di massima si osservano percentuali d'ordine di poco inferiori di quelle ottenute nel test precedente e come sempre l'imprevedibilità dell'intensità del disturbo non permette conclusioni precise circa le percentuali di fallimento.

Capitolo 6

Conclusioni

Ricapitolando, in questo lavoro di tesi si sono analizzate le prestazioni delle comunicazioni powerline effettuate mediante i chip Cypress CY8CPLC20 in Banda CENELEC-C (frequenze attorno ai 130kHz, baudrate 2400 bps). Per farlo, dopo aver studiato il funzionamento dei chip, si sono realizzate delle specifiche applicazioni per il rilevamento delle misure. In particolare si sono realizzati programmi per il chip (mediante software proprietario) che interagiscono con un programma Windows (ad interfaccia grafica realizzato in Visual Basic) mediante cavo seriale, per svolgere esperimenti di misure riguardanti i tempi e gli esiti delle trasmissioni powerline di vario tipo:

- senza protocollo di rete, inviando byte
- con protocollo di rete, inviando pacchetti in modalità con o senza acknowledgement
- con protocollo di rete, realizzando una comunicazione di tipo polling

Un'analisi approfondita dei tempi rilevati ha permesso di determinare l'effettiva velocità di trasmissione dei bit ed il ritardo che si ha tra i vari byte inviati ottenendo risultati in accordo con i valori dichiarati dal costruttore. Dopo aver individuato i vari modelli che permettono di stimare i tempi di trasmissione si è proceduto determinandone i parametri. Si è evidenziata una discordanza in termini di ritardo tra i byte tra il metodo di invio dei dati con il protocollo di rete Cypress e quello senza.
Dopo aver constatato l'efficienza delle comunicazioni in una rete priva di disturbi, si è proceduto studiando le percentuali di fallimento delle trasmissioni quando alla rete viene connesso un inverter collegato a dei pannelli solari che introduce un disturbo. Non si sono potute identificare in modo preciso le conseguenze poiché si è appurato che il disturbo stesso variava di molto ed imprevedibilmente nel tempo. Si sono tuttavia osservate percentuali di errore (sia di trasmissioni pacchetto-ack che di polling) distribuite principalmente attorno a valori del 2/3% totali nei casi migliori, con molta variabilità a seconda della lunghezza del payload del pacchetto, della posizione dei dispositivi rispetto alla fonte del disturbo e delle sensibilità di ricezione utilizzate. Il valore del $Packet\ Error\ Rate$ dei pacchetti di andata invece si distribuisce principalmente attorno all'1%, con molte eccezioni lontane da questo valore specialmente in un caso critico.

Solamente una delle configurazioni osservate infatti ha portato sempre a risultati notevolmente peggiori con percentuali di errore che passavano dal 5%, al 20% fino al 40%. Non si è riusciti però a stabilirne la causa in quanto altre configurazioni, identiche ma con i modem disposti in nodi diversi della rete, hanno portano a risultati completamente differenti, rendendo impossibile la determinazione dell'origine del problema. Si è osservato infine che una regolazione dei parametri, specialmente della soglia di ricezione, permette comunque leggeri miglioramenti.
Nell'ottica di valutarne l'effettiva applicabilità in campo domotico, quando è stato possibile si sono misurati i tempi che si ottengono imponendo la ritrasmissione dei dati fino alla ricezione dell'Ack. Questo, ad esempio per trasmissioni di tipo pacchetto-ack con 31 byte di payload, ha portato a tempi medi di circa 324 ms e massimi dell'ordine del secondo, contro i regolari 259 ms. Se si rispettano inoltre le normative CENELEC di accesso CSMA al mezzo, i tempi medi passano a circa 457 ms mentre quelli massimi raddoppiano.
Si può facilmente comprendere che tempi di risposta così lunghi potrebbero essere accettabili per determinate applicazioni ma non per tutte. Si pensi ad esempio rispettivamente alla regolazione della temperatura di un termosifone (che non richiede dinamiche elevate) e all'accensione di una luce (per la quale tempi di risposta di quasi 2 secondi sono decisamente troppo alti).
Non avendo identificato il disturbo e non potendolo confrontare con quello immesso nella rete dagli inverter commerciali che si potrebbero trovare in ambienti domestici, si è obbligati a concludere che non è possibile garantire un'applicabilità globale per questo tipo di comunicazioni in ambito domotico, senza dimenticare inoltre che non è stata affrontata l'analisi dei problemi causati dalla connessione e disconnessione dei carichi dalla rete che potrebbero presumibilmente disturbare anche le frequenze utilizzate dalle PLC.

Appendici

Appendice A

Chip CY8CPLC20

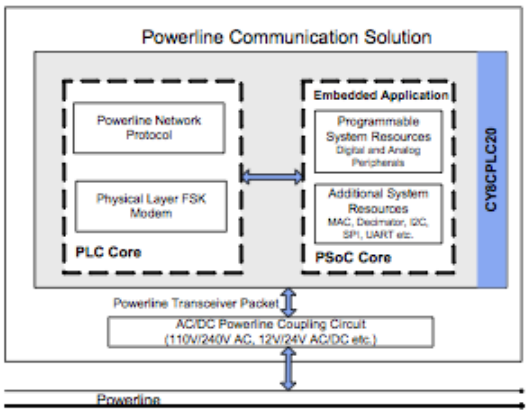
CY8CPLC20

Powerline Communication Solution

Features

- Powerline communication solution
  - Integrated powerline modem PHY
  - Frequency shift keying modulation
  - Configurable baud rates up to 2400 bps
  - Powerline optimized network protocol
  - Integrates data link, transport, and network layers
  - Supports bidirectional half duplex communication
  - 8-bit CRC error detection to minimize data loss
  - I^2C enabled powerline application layer
  - Supports I^2C frequencies of 50, 100, and 400 kHz
  - Reference designs for 110 V/240 V AC and 12 V/24 V AC/DC Powerlines
  - Reference designs comply with CENELEC EN 50065-1:2001 and FCC Part 15
- Powerful Harvard-architecture Processor
  - M8C processor speeds to 24 MHz
  - Two 8x8 multiply, 32-bit accumulate
- Programmable system resources (PSoC® Blocks)
  - 12 Rail-to-Rail Analog PSoC Blocks provide:
    - Up to 14-bit ADCs
    - Up to 9-bit DACs
    - Programmable gain amplifiers
    - Programmable filters and comparators
  - 16 Digital PSoC Blocks provide:
    - 8 to 32-bit Timers, Counters, and PWMs
    - CRC and PRS Modules
    - Up to four full duplex UARTs
    - Multiple SPI™ masters or slaves
    - Connectable to all GPIO Pins
  - Complex peripherals by combining blocks
- Flexible on-chip memory
  - 32 KB flash program storage 50,000 erase or write cycles
  - 2 KB SRAM data storage
  - EEPROM emulation in flash
- Programmable pin configurations
  - 25 mA sink, 10 mA source on all GPIOs
  - Pull-up, Pull-down, high Z, strong, or open drain drive Modes on all GPIO
  - Up to 12 analog inputs on all GPIOs
  - Configurable interrupt on all GPIOs
- Additional system resources
  - I^2C slave, master, and multi-master to 400 kHz
  - Watchdog and sleep timers
  - User-configurable low-voltage detection
  - Integrated supervisory circuit
  - On-chip precision voltage reference
- Complete development tools
  - Free development software (PSoC Designer™)
  - Full-featured in-circuit emulator (ICE) and programmer
  - Full-speed emulation
  - Complex breakpoint structure
  - 128 KB trace memory
  - Complex events
  - C Compilers, assembler, and linker

Logic Block Diagram



Cypress Semiconductor Corporation • 198 Champion Court • San Jose, CA 95134-1709 • 408-943-2600
Document Number: 001-48325 Rev. *L Revised September 11, 2014

Figura A.1: *Prima facciata del Datasheet del Chip CY8PLC20*

Cmd ID	Command Name	Description	Payload (TX Data)	Response (RX Data)
0x01	SetRemote_TXEnable	Sets the TX Enable bit in the PLC Mode Register. Rest of the PLC Mode register is unaffected	0 - Disable Remote TX 1 - Enable Remote TX	If Remote Lock Config = 0, Response = 00 (Success) If Remote Lock Config = 1, Response = 01 (Denied)
0x03	SetRemote_ExtendedAddr	Set the Addressing to Extended Addressing Mode	0 - Disable Extended Addressing 1 - Enable Extended Addressing	If Remote Lock Config = 0, Response = 00 (Success) If Remote Lock Config = 1, Response = 01 (Denied)
0x04	SetRemote_LogicalAddr	Assigns the specified logical address to the remote PLC node	If Ext Address = 0, Payload = 8-bit Logical Address If Ext Address = 1, Payload = 16-bit Logical Address	If Remote Lock Config = 0, Response = 00 (Success) If Remote Lock Config = 1, Response = 01 (Denied)
0x05	GetRemote_LogicalAddr	Get the Logical Address of the remote PLC node	None	If Remote TX Enable = 0, Response = None If Remote TX Enable = 1, {If Ext Address = 0, Response = 8-bit Logical Address If Ext Address = 1, Response = 16-bit Logical Address}
0x06	GetRemote_PhysicalAddr	Get the Physical Address of the remote PLC node	None	If Remote TX Enable = 0, Response = None If Remote TX Enable = 1, Response = 64-bit Physical Address
0x07	GetRemote_State	Request PLC_Mode Register content from a Remote PLC node	None	If Remote TX Enable = 0, Response = None If Remote TX Enable = 1, Response = Remote PLC Mode register
0x08	GetRemote_Version	Get the Version Number of the Remote Node	None	If TX Enable = 0, Response = None If TX Enable = 1, Response = Remote Version register
0x09	SendRemote_Data	Transmit data to a Remote Node.	Payload = Local TX Data	If Local Service Type = 0, Response = None If Local Service Type = 1, Response = Ack
0x0A	RequestRemote_Data	Request data from a Remote Node	Payload = Local TX Data	If Local Service Type = 1, Response = Ack Then, the remote node host must send a ResponseRemote_Data command. The response must be completely transmitted within 1.5s of receiving the request. Otherwise, the requesting node will time out.
0x0B	ResponseRemote_Data	Transmit response data to a Remote Node.	Payload = Local TX Data	None
0x0C	SetRemote_BIU	Enables/Disables BIU functionality at the remote node	0 - Enable Remote BIU 1 - Disable Remote BIU	If Remote Lock Config = 0, Response = 00 (Success) If Remote Lock Config = 1, Response = 01 (Denied)
0x0D	SetRemote_ThresholdValue	Sets the Threshold Value at the Remote node	3-bit Remote Threshold Value	If Remote Lock Config = 0, Response = 00 (Success) If Remote Lock Config = 1, Response = 01 (Denied)
0x0E	SetRemote_GroupMembership	Sets the Group Membership of the Remote node	Byte0 - Remote SIngle Group Membership Address Byte1- Remote Multiple Group Membership Address	If Remote Lock Config = 0, Response = 00 (Success) If Remote Lock Config = 1, Response = 01 (Denied)
0x0F	GetRemote_GroupMembership	Gets the Group Membership of the Remote node	None	If Remote TX Enable = 0, Response = None If Remote TX Enable = 1, Response = Byte0 - Remote SIngle Group Membership Address Byte1- Remote Multiple Group Membership Address
0x10–0x2F	Reserved			

Figura A.2: *Comandi remoti disponibili utilizzando il protocollo di rete Cypress*

Offset	Register Name	I²C Access[a]	7	6	5	4	3	2	1	0
0x00	INT_Enable	RW	INT_Clear	INT_Polarity	INT_UnableToTX	INT_TX_NO_ACK	INT_TX_NO_RESP	INT_RX_Packet_Dropped	INT_RX_Data_Available	INT_TX_Data_Sent
0x01	Local_LA_LSB	RW	8-bit Logical Address / LSB for extended 16-bit address							
0x02	Local_LA_MSB	RW	MSB for 16-bit Extended Address							
0x03	Local_Group	RW	8-bit Group Address							
0x04	Local_Group_Hot	RW	One Hot Encoded (for example. if byte = 0b00010001, then member of groups #5 and #1)							
0x05	PLC_Mode	RW	TX_Enable	RX_Enable	Lock_Configuration	Disable_BIU	RX_Override	Set_Ext_Address	Promiscuous_MASK	Promiscuous_CRC_MASK
0x06	TX_Message_Length	RW	Send_Message	Reserved		Payload_Length_MASK				
0x07	TX_Config	RW	TX_SA_Type	TX_DA_Type		TX_Service_Type	TX_Retry			
0x08	TX_DA	RW	Remote Node Destination Address (8 bytes)							
0x10	TX_CommandID	RW	TX Command ID							
0x11	TX_Data	RW	TX Data (31 bytes)							
0x30	Threshold_Noise	RW	Reserved	Auto_BIU_Threshold	Reserved		BIU_Threshold_Constant			
0x31	Modem_Config	RW	Reserved	Modem_TXDelay		Reserved	Modem_FSKBW	Reserved	Modem_BPS	
0x32	TX_Gain	RW	Reserved				TX_Gain_Mask			
0x33	RX_Gain	RW	Reserved					RX_Gain_Mask		
0x34	Timing_Config	RW	BIU_Interval_Min		BIU_Interval_Span		BIU_Timeout	Reserved	ACK_Timeout	
0x35-0x3F	Reserved	RW	Reserved							
0x40	RX_Message_INFO	RW	New_RX_Msg	RX_DA_Type	RX_SA_Type	RX_Msg_Length				
0x41	RX_SA	R	Remote Node Source Address(8 Bytes)							
0x49	RX_CommandID	R	RX Command ID							
0x4a	RX_Data	R	RX Data (31 bytes)							
0x69	INT_Status	R	Status_Value_Change	Reserved	Status_UnableToTX	Status_TX_NO_ACK	Status_TX_NO_RESP	Status_RX_Packet_Dropped	Status_RX_Data_Available	Status_TX_Data_Sent
0x6A	Local_PA	R	Physical Address (8 bytes), "0x6A -> MSB"							
0x72	Local_FW	R	Version Number							

Figura A.3: *Registri disponibili utilizzando il protocollo di rete Cypress*

Appendice B

Statistiche trasmissioni senza protocollo di rete

B.1 Tempi senza inverter

Byte Inviati	Tempo TX [ms]	Byte RX	Tempo RX [ms]
250	1287,21875	250	1275,09375
250	1286,8125	250	1275,46875
250	1287,21875	250	1275,59375
250	1287,21875	250	1275,53125
250	1287,21875	250	1275,53125
200	1031,0625	200	1019,78125
200	1031,0625	200	1019,78125
200	1031,46875	200	1019,8125
200	1031,0625	200	1019,8125
200	1031,46875	200	1019,75
150	775,34375	150	764,09375
150	775,34375	150	764,0625
150	775,3125	150	764,125
150	775,71875	150	763,96875
150	775,71875	150	764,0625
100	520	100	508,3125
100	519,59375	100	508,3125
100	520	100	508,375
100	520	100	508,3125
100	520	100	508,28125
50	264,25	50	252,5625
50	263,875	50	252,59375
50	263,84375	50	252,59375
50	264,28125	50	252,5625
50	263,875	50	252,625
10	59,65625	10	47,96875
10	59,25	10	48
10	59,96875	10	47,96875
10	59,25	10	47,96875
10	59,96875	10	48
1	13,65625	1	/
1	13,25	1	/
1	13,65625	1	/
1	13,25	1	/
1	13,25	1	/

Tabella B.1: *Tempi di invio e ricezione, clock a 24 MHz*

	Tempo di trasmissione [ms]			
Byte Inviati	**24MHz**	**24MHz/4**	**24MHz/8**	**24MHz/32**
250	1287,21875	1288,625	1290,875	1418,25
250	1286,8125	1288,625	1290,90625	1418,25
250	1287,21875	1288,5625	1290,90625	1418,1875
250	1287,21875	1288,5625	1290,875	1418,21875
250	1286,8125	1288,625	1290,90625	1418,25
200	1287,21875	1288,59375	1290,90625	1418,25
200	1031,46875	1032,84375	1035,09375	1141,21875
200	1031,0625	1032,875	1035,125	1141,1875
200	1031,46875	1032,875	1035,09375	1141,1875
200	1031,46875	1032,84375	1035,0625	1141,21875
150	1031,09375	1032,84375	1035,09375	1141,4375
150	775,34375	777,125	779,375	864,125
150	775,3125	777,15625	779,375	864,09375
150	775,34375	777,15625	779,34375	864,1875
150	775,71875	777,125	779,4375	864,09375
100	519,59375	521,40625	523,6875	587,125
100	520	521,375	523,71875	587,09375
100	520,03125	521,40625	523,6875	587,09375
100	520	521,40625	523,65625	587,09375
50	519,625	521,40625	523,6875	587,09375
50	263,875	265,65625	267,96875	310,0625
50	263,875	265,65625	267,96875	310,125
50	264,28125	265,65625	267,96875	310,09375
50	264,25	265,65625	267,96875	310
10	263,84375	265,65625	267,9375	310,09375
10	59,28125	61,0625	63,3125	88,40625
10	59,25	61,0625	63,28125	88,40625
10	59,6875	61,0625	63,3125	88,375
10	59,6875	61,03125	63,3125	88,5625
1	59,65625	61,0625	63,3125	88,40625
1	13,25	15,03125	17,28125	38,5625
1	13,25	15,03125	17,28125	38,5
1	13,625	15,03125	17,28125	38,5625
1	13,25	15	17,25	38,53125

Tabella B.2: *Tempi di invio, clock a varie frequenze*

Appendice C

Statistiche trasmissioni con protocollo di rete

C.1 Esiti delle trasmissioni

Nella presente sezione vengono illustrati i principali risultati dei test in termini di successo delle trasmissioni. Si ricorda che le impostazioni del trasmettitore coincidono sempre con quelle del ricevitore. Nelle seguenti tabelle:
”Da” indica la posizione del dispositivo trasmettitore, in accordo con lo schema della rete di Figura 3.14
”A” indica la posizione del dispositivo ricevitore
”Inverter” indica la posizione dell’inverter
”Ack” indica il numero di ack presenti nella trasmissione
”RXGain” indica la soglia di ricezione
”TXGain” indica il guadagno di trasmissione
”BIU” indica se è attiva la BIU detestino, quindi l’accesso CSMA al mezzo.
”Retry” indica il valore di retry
”TXDelay” indica la durata del TXDelay impostata
”Timestamp” indica la data e l’ora della prova
”%fall” indica la percentuale di fallimenti delle trasmissioni
”Successi” indica il numero di trasmissioni complete rispetto al numero di trasmissioni totali
”RX” indica il numero di pacchetti correttamente ricevuti dal ricevitore
”PER” indica il $Packet\ Error\ Rate$ relativo ai pacchetti di sola andata

Da	A	RXGain	TXGain [V]	BIU	Retry	TXDelay [ms]	Timestamp	% fall.	Successi	PER
1	6	250 uV	1.55	Off	0	7	2015/07/22 14:15:45	0	5000/5000	5000
1	6	250 uV	1.55	Off	0	7	2015/07/29 13:17:43	0	5000/5000	5000
2	6	250 uV	1.55	Off	0	7	2015/07/29 13:44:27	0	5000/5000	5000
3	6	250 uV	1.55	Off	0	7	2015/07/29 12:58:43	0	5000/5000	5000
4	6	250 uV	1.55	Off	0	7	2015/07/22 13:55:44	0	5000/5000	5000

Tabella C.1: *Percentuali di fallimento pacchetto-ack, senza inverter nella rete*

Da	A	Ack	RXGain	TXGain [V]	BIU	Retry	TXDelay [ms]	Timestamp	% fall.	Successi	RX	PER
1	1	No	250 uV	1.55	Off	0	7	2015/09/28 13:20:17	0.94	4953/5000	4963	0.74
1	1	No	1.25V	1.55	Off	0	7	2015/09/28 13:51:36	0.82	4959/5000	4976	0.48
1	2	No	250 uV	1.55	On	0	7	2015/09/28 15:39:07	0.26	4987/5000	4998	0.04
1	2	No	250 uV	1.55	Off	0	7	2015/09/28 10:50:01	0.58	4971/5000	4982	0.36
1	2	No	250 uV	1.55	Off	0	7	2015/07/22 17:16:13	1.08	4946/5000	4971	0.58
1	2	No	250 uV	1.55	Off	0	7	2015/09/16 10:06:36	0.76	4962/5000	4977	0.46
1	2	No	250 uV	1.55	Off	0	25	2015/09/28 16:09:41	0.58	4971/5000	4977	0.46
1	2	2	250 uV	1.55	Off	0	7	2015/07/15 16:54:27	2.1	4895/5000	4915	1.7
1	2	2	250 uV	1.55	Off	0	7	2015/07/2216:49:57	0.78	4961/5000	4976	0.48
1	2	No	1.25 V	1.55	Off	0	7	2015/09/28 11:19:03	1.12	4944/5000	4963	0.74
1	2	No	1.25 V	1.55	Off	0	7	2015/09/2814:22:16	1.16	4942/5000	4966	0.68
1	2	No	1.25 V	3.50	Off	0	7	2015/09/28 14:52:51	0.72	4964/5000	4974	0.52
1	3	No	1.25 V	1.55	Off	0	7	2015/09/28 11:48:11	1.08	4946/5000	4966	0.68
1	6	No	250 uV	1.55	Off	0	7	2015/09/28 12:51:17	1.32	4934/5000	4952	0.96
1	6	2	250 uV	1.55	Off	0	7	2015/07/28 10:21:25	0.96	4952/5000	4982	0.36
1	6	No	1.25 V	1.55	Off	0	7	2015/09/28 12:18:56	0.72	4964/5000	4977	0.72
2	3	No	250 uV	1.55	Off	0	7	2015/09/16 10:33:27	0.9	4955/5000	4975	0.5
2	6	No	250 uV	1.55	Off	0	7	2015/09/16 11:01:08	0.7	4965/5000	4983	0.34
2	6	No	250 uV	1.55	Off	0	25	2015/09/16 12:32:52	1.26	4937/5000	4959	0.82
2	6	2	250 uV	1.55	Off	5	7	2015/09/16 13:13:56	0	5000/5000	5000	0
2	6	2	250 uV	1.55	On	5	7	2015/09/16 14:05:00	0.2	4990/5000	5000	0
2	6	No	5 mV	3.5	Off	0	7	2015/09/16 12:01:35	1.58	4921/5000	4942	1.16

Tabella C.2: *Percentuali di fallimento test di polling, senza inverter nella rete*

Da	A	Inverter	RXGain	TXGain [V]	BIU	Retry	TXDelay [ms]	Timestamp	% fall.	Successi	RX	PER
1	2	6	250 uV	1.55 V	Off	0	7	2015/07/09 16-21-35	0	5000/5000	5000	0
1	3	6	250 uV	1.55 V	Off	0	7	2015/07/09 16:41:45	0	5000/5000	5000	0
1	4	6	250 uV	1.55 V	Off	0	7	2015/07/10 17:26:12	0	5000/5000	5000	0
1	6	6	250 uV	1.55 V	Off	0	7	2015/07/09 13:36:35	21.12	3944/5000	4112	17.76
1	6	6	250 uV	1.55 V	Off	0	7	2015/07/09 16:02:39	22.66	3867/5000	4072	18.56
1	6	6	250 uV	1.55 V	Off	0	7	2015/07/10 12:05:04	3.8	4810/5000	4823	3.53
1	6	6	250 uV	1.55 V	Off	0	7	2015/07/10 14:19:33	3.88	4806/5000	4822	3.56
1	6	6	250 uV	1.55 V	Off	0	7	2015/07/10 16:54:01	20.78	3961/5000	4115	17.7
1	6	6	250 uV	1.55 V	Off	0	7	2015/07/28 11:11:11	3.2	4840/5000	4843	3.14
1	6	6	250 uV	1.55 V	Off	0	7	2015/07/28 11:33:05	11.1	4445/5000	4492	10.16
1	6	6	250 uV	1.55 V	Off	0	7	2015/07/28 12:21:21	19	4050/5000	4189	16.22
1	6	6	250 uV	1.55 V	Off	0	7	2015/07/28 12:39:45	19.62	4019/5000	4177	16.46
1	6	6	250 uV	1.55 V	Off	0	7	2015/07/28 14:39:01	25.2	3740/5000	3979	20.42
1	6	6	250 uV	1.55 V	Off	0	7	2015/07/29 11:34:35	20.96	3952/5000	4097	18.06
1	6	6	250 uV	1.55 V	On	3	7	2015/07/09 17:43:56	0.1	4995/5000	4998	0.04
1	6	6	250 uV	1.55 V	Off	3	7	2015/07/09 15:05:31	0.42	4979/5000	5113	/
1	6	6	1.25 V	1.55 V	Off	0	7	2015/09/17 18:18:25	9.06	4547/5000	4568	8.64
1	6	6	1.25 V	1.55 V	Off	0	7	2015/09/17 16:28:10	11.88	4406/5000	4437	11.26
1	6	6	1.25 V	1.55 V	Off	5	7	2015/09/17 17:08:56	0	5000/5000	5000	0
1	6	6	1.25 V	1.55 V	On	5	7	2015/09/17 17:42:30	0	5000/5000	5000	0
1	6	6	5 mV	3.50 V	Off	0	7	2015/07/09 15:42:08	12.92	4354/5000	4355	12.9
2	1	1	250 uV	1.55 V	Off	0	7	2015/07/29 10:36:33	0	5000/5000	5000	0
2	3	3	250 uV	1.55 V	Off	0	7	2015/07/29 12:11:46	6.4	4680/5000	4749	5.02
2	4	6	250 uV	1.55 V	Off	0	7	2015/07/09 17:07:35	1.28	4936/5000	4936	1.28
2	6	6	250 uV	1.55 V	Off	0	7	2015/07/09 13:55:18	20.52	3974/5000	4103	17.94
2	6	6	250 uV	1.55 V	Off	0	7	2015/07/28 15:00:38	23.08	3846/5000	4023	19.54
3	6	6	250 uV	1.55 V	Off	0	7	2015/07/09 13:17:46	0.84	4958/5000	4974	0.52
3	6	6	250 uV	1.55 V	Off	0	7	2015/07/09 14:14:03	1.36	4932/5000	4960	0.8
3	6	6	250 uV	1.55 V	Off	0	7	2015/07/28 12:03:00	0.48	4976/5000	4984	0.32
3	6	6	1.25 V	1.55 V	Off	0	7	2015/09/17 17:59:46	0.52	4974/5000	4978	0.44
4	1	1	250 uV	1.55 V	Off	0	7	2015/07/29 10:56:16	0	5000/5000	5000	0
4	6	6	250 uV	1.55 V	Off	0	7	2015/07/09 14:38:42	1.3	4935/5000	4949	1.08
6	1	1	250 uV	1.55 V	Off	0	7	2015/07/29 11:15:16	0	5000/5000	5000	0
6	1	6	1.25 V	1.55 V	Off	0	7	2015/09/17 16:46:58	12.9	4355/5000	4918	1.64
6	2	2	250 uV	1.55 V	Off	0	7	2015/07/29 12:39:15	4.36	4782/5000	4782	4.36

Tabella C.3: *Percentuali di fallimento pacchetto-ack, con inverter nella rete*

Da	A	Inverter	Ack	RXGain	TXGain [V]	BIU	Retry	TXDelay [ms]	Timestamp	% fall.	Successi	RX	PER
1	2	6	No	250 uV	1.55 V	Off	0	7	2015/07/28 17/02/14	0.9	4955/5000	4976	0.48
1	6	6	No	250 uV	1.55 V	Off	0	7	2015/07/28 14:12:07	19.34	4033/4926	4460	10.8
1	6	6	2	250 uV	1.55 V	Off	5	7	2015/07/28 13:29:08	21.56	3922/5000	4298	14.04
1	6	6	No	1.25 V	1.55 V	Off	5	7	2015/09/17 13:36:49	4.18	4791/5000	4809	3.82
1	6	6	2	1.25 V	1.55 V	Off	5	7	2015/09/17 15:40:14	4.32	4784/5000	4957	0.86
1	6	6	No	5 mV	1.55 V	Off	0	7	2015/09/17 14:55:37	43.36	2832/5000	2901	41.98
2	1	1	No	250 uV	1.55 V	Off	0	7	2015/07/29 10:07:45	1.08	4946/5000	4974	0.52
2	6	6	No	250 uV	1.55 V	Off	0	7	2015/07/28 15:47:12	18.529	3755/4609	4336	13.28
2	6	6	2	250 uV	1.55 V	Off	0	7	2015/07/28 16:31:43	21.44	3928/5000	4269	14.62
3	6	6	No	1.25 V	1.55 V	Off	0	7	2015/09/17 13:08:39	0.3	4985/5000	4993	0.14
3	6	6	No	250 uV	1.55 V	Off	0	7	2015/07/28 17:29:43	4.9	4755/5000	4958	0.84
6	1	1	No	1.25 V	1.55 V	Off	0	7	2015/09/17 14:03:37	0.68	4966/5000	4966	0.68
6	6	6	No	1.25 V	1.55 V	Off	0	7	20150917124344	1.02	4949/5000	4968	0.64

Tabella C.4: *Percentuali di fallimento test di polling, con inverter nella rete*

C.2 Tempi delle trasmissioni

In questa sezione vengono presentati i principali risultati relativi ai tempi di trasmissione di alcune prove, in particolare "Tmin", "Tmax" e "Tmed" indicano rispettivamente il tempo minore, maggiore e medio relativi alle sole trasmissioni con esito positivo o negativo a seconda della riga. Quando il valore di $retry$ non è specificato si intende pari a zero.

Byte Payload	Esito	%	Tmin [ms]	Tmax [ms]	Tmed [ms]
Tempo di sola trasmissione (senza ack)					
0	Successo	100	51.062	51.219	51.1392
0	Fallimento	0	/	/	/
7	Successo	100	87.031	87.188	87.1046
7	Fallimento	0	/	/	/
15	Successo	100	128.156	128.25	128.2023
15	Fallimento	0	/	/	/
23	Successo	100	169.25	169.406	169.3248
23	Fallimento	0	/	/	/
31	Successo	100	210.375	210.469	210.4112
31	Fallimento	0	/	/	/
Da 1 a 6, BIUOff, TXDelay=7ms, 2015/07/22 14:15:45					
0	Successo	100	99.094	99.375	99.2064
0	Fallimento	0	/	/	/
7	Successo	100	135.219	135.469	135.3218
7	Fallimento	0	/	/	/
15	Successo	100	176.469	176.781	176.6133
15	Fallimento	0	/	/	/
23	Successo	100	217.75	218.031	217.8898
23	Fallimento	0	/	/	/
31	Successo	100	259	259.312	259.1549
31	Fallimento	0	/	/	/
Da 1 a 6, BIUOff, TXDelay=7ms, 2015/07/29 13:17:43					
0	Successo	100	99.094	99.375	99.2117
0	Fallimento	0	/	/	/
7	Successo	100	135.219	135.469	135.3314
7	Fallimento	0	/	/	/
15	Successo	100	176.5	176.781	176.6282
15	Fallimento	0	/	/	/
23	Successo	100	217.75	218.062	217.9104
23	Fallimento	0	/	/	/
31	Successo	100	259.031	259.312	259.1708
31	Fallimento	0	/	/	/

Tabella C.5: *Analisi dei principali Log di Trasmissione senza inverter nella rete (quando non specificato si intende con ack)*

Byte Payload	Esito	%	Tmin [ms]	Tmax [ms]	Tmed [ms]
Da 1 a 2, No Ack, BIUOff, TXDelay=7ms, 2015/07/22 17:16:13					
0	Successo	99.6	101.938	102.5	102.2246
0	Fallimento	0.4	1000.031	1000.125	1000.0833
7	Successo	99.4	174.188	174.812	174.4602
7	Fallimento	0.6	1000.062	1000.219	1000.1012
15	Successo	99.2	256.719	257.312	256.9891
15	Fallimento	0.8	1000.062	1000.219	1000.1211
23	Successo	98.6	339.188	339.844	339.5638
23	Fallimento	1.4	1000.031	1000.219	1000.1339
31	Successo	97.8	421.75	422.375	422.0847
31	Fallimento	2.2	1000.031	1000.219	1000.1178
Da 2 a 3, No Ack, BIUOff, TXDelay=7ms, 2015/09/16 10:33:27					
0	Successo	99.7	101.906	102.531	102.2236
0	Fallimento	0.3	1000.062	1000.219	1000.1457
7	Successo	99.7	174.156	174.719	174.4402
7	Fallimento	0.3	1000.094	1000.219	1000.1563
15	Successo	98.7	256.719	257.312	257
15	Fallimento	1.3	1000.031	1000.219	1000.1347
23	Successo	99.1	339.219	339.844	339.5429
23	Fallimento	0.9	1000.031	1000.25	1000.1249
31	Successo	98.3	421.844	422.375	422.108
31	Fallimento	1.7	1000.031	1000.219	1000.1415
Da 2 a 6, No Ack, BIU Off, TXDelay=25 ms, 2015/09/16 12:32:52					
0	Successo	99.5	145.406	146.031	145.6982
0	Fallimento	0.5	1000.125	1000.25	1000.1624
7	Successo	98.8	217.656	218.219	217.9282
7	Fallimento	1.2	1000.031	1000.25	1000.1485
15	Successo	98.8	300.125	300.781	300.4727
15	Fallimento	1.2	1000.062	1000.156	1000.1065
23	Successo	98.3	382.75	383.312	383.024
23	Fallimento	1.7	1000.031	1000.25	1000.1562
31	Successo	98.3	465.281	465.844	465.5694
31	Fallimento	1.7	1000.031	1000.25	1000.1267
Da 2 a 6, 2 ack, 5 retry, BIU Off, TXDelay=7ms, 2015/09/16 13:13:56					
0	Successo	100	203.906	325.594	204.6264
0	Fallimento	0	/	/	/
7	Successo	100	276.062	435.688	277.9167
7	Fallimento	0	/	/	/
15	Successo	100	358.719	559.156	361.7854
15	Fallimento	0	/	/	/
23	Successo	100	441.219	684.406	443.6705
23	Fallimento	0	/	/	/
31	Successo	100	523.688	809.906	527.1396
31	Fallimento	0	/	/	/
Da 2 a 6, 2 ack, 5 retry, BIU On, TXDelay=7ms, 2015/09/16 14:05:00					
0	Successo	100	384.344	639.844	409.6609
0	Fallimento	0	/	/	/
7	Successo	100	460.594	766.125	483.8395
7	Fallimento	0	/	/	/
15	Successo	100	539.219	877.719	566.2877
15	Fallimento	0	/	/	/
23	Successo	100	621.75	1003.188	650.0105
23	Fallimento	0	/	/	/
31	Successo	99	329.969	1089.219	726.2774
31	Fallimento	1	1000.062	1000.219	1000.1626

Tabella C.6: *Analisi dei principali Log di Trasmissione senza inverter nella rete, polling*

Byte Payload	Esito	%	Tmin [ms]	Tmax [ms]	Tmed [ms]
Da 1 a 2, inverter in 6, BIU Off, TXDelay=7ms, 2015/07/09 16:21:35					
0	Successo	100	99.062	99.344	99.2059
0	Fallimento	0	/	/	/
7	Successo	100	135.188	135.469	135.3211
7	Fallimento	0	/	/	/
15	Successo	100	176.469	176.781	176.6192
15	Fallimento	0	/	/	/
23	Successo	100	217.719	218.031	217.8929
23	Fallimento	0	/	/	/
31	Successo	100	259	266.562	259.1611
31	Fallimento	0	/	/	/
Da 1 a 6, inverter in 6, BIU Off, TXDelay=7 ms, 2015/07/10 12:05:04					
0	Successo	98.9	99.094	99.406	99.2517
0	Fallimento	1.1	121.562	121.625	121.5852
7	Successo	98.5	135.219	135.5	135.3604
7	Fallimento	1.5	159.531	159.594	159.5497
15	Successo	97.3	176.531	176.844	176.6752
15	Fallimento	2.7	200.656	200.75	200.682
23	Successo	95.5	217.781	218.062	217.9222
23	Fallimento	4.5	243.719	243.844	243.7659
31	Successo	90.8	259.031	259.344	259.1918
31	Fallimento	9.2	286.812	286.938	286.8832
Da 1 a 6, inverter in 6, BIU Off, TXDelay=7 ms, 2015/07/09 16:02:39					
0	Successo	86.8	99.094	99.406	99.2696
0	Fallimento	13.2	121.562	121.656	121.6073
7	Successo	81.7	135.188	135.562	135.3797
7	Fallimento	18.3	159.531	159.625	159.572
15	Successo	81.3	176.469	176.781	176.6484
15	Fallimento	18.7	200.625	200.719	200.6668
23	Successo	77	217.812	218.094	217.9483
23	Fallimento	23	243.719	243.875	243.787
31	Successo	59.9	259.062	266.5	259.2039
31	Fallimento	40.1	286.781	286.969	286.8872
Da 1 a 6, inverter in 6, 5 retry, BIU Off, TXDelay=7 ms, 2015/09/17 17:08:56					
0	Successo	100	99.094	463.031	115.742
0	Fallimento	0	/	/	/
7	Successo	100	135.188	612.5	160.4856
7	Fallimento	0	/	/	/
15	Successo	100	176.5	776.531	200.6559
15	Fallimento	0	/	/	/
23	Successo	100	217.812	1189.344	252.1712
23	Fallimento	0	/	/	/
31	Successo	100	259.031	1402.312	323.486
31	Fallimento	0	/	/	/
Da 1 a 6, inverter in 6, 5 retry, BIU On, TXDelay=7 ms, 20150917174230					
0	Successo	100	189.344	879.812	231.7505
0	Fallimento	0	/	/	/
7	Successo	100	225.375	1009.312	279.5632
7	Fallimento	0	/	/	/
15	Successo	100	266.719	1271.406	343.6341
15	Fallimento	0	/	/	/
23	Successo	100	1.625	1665.438	403.0429
23	Fallimento	0	/	/	/
31	Successo	100	349.188	2039.469	457.2952
31	Fallimento	0	/	/	/

Tabella C.7: *Analisi dei principali Log di Trasmissione con inverter nella rete, pacchetto-ack*

Byte Payload	Esito	%	Tmin [ms]	Tmax [ms]	Tmed [ms]
Da 1 a 6, inverter in 6, No Ack, BIU Off, TXDelay=7 ms, 2015/09/17 13:36:49					
0	Successo	98.3	102	102.5	102.2362
0	Fallimento	1.7	1000.031	1000.219	1000.1139
7	Successo	98.3	174.188	174.719	174.4578
7	Fallimento	1.7	1000.031	1000.219	1000.1342
15	Successo	96	256.688	257.312	257.0208
15	Fallimento	4	1000.031	1000.219	1000.1352
23	Successo	94	339.281	339.844	339.5616
23	Fallimento	6	1000.031	1000.219	1000.1261
31	Successo	92.5	416.938	422.406	422.1153
31	Fallimento	7.5	1000.031	1000.219	1000.1183
Da 2 a 6, inverter in 6, 2 Ack, BIU Off, TXDelay=7 ms, 2015/07/28 16:31:43					
0	Successo	86.6	203.906	204.5	204.182
0	Fallimento	13.4	1000.031	1000.25	1000.1182
7	Successo	85.3	276.094	276.719	276.3908
7	Fallimento	14.7	1000.031	1000.281	1000.1331
15	Successo	75.6	358.719	359.281	359.016
15	Fallimento	24.4	1000.031	1000.25	1000.1356
23	Successo	73.2	436.5	441.844	441.5373
23	Fallimento	26.8	1000.031	1000.281	1000.1331
31	Successo	72.1	523.719	524.406	524.0674
31	Fallimento	27.9	1000.031	1000.281	1000.1346
Da 1 a 6, inverter in 6, 2 Ack, 5 retry, BIU Off, TXDelay=7 ms, 2015/09/17 15:40:14					
0	Successo	99.9	203.875	843	227.0935
0	Fallimento	0.1	1000.125	1000.125	1000.125
7	Successo	99	276.094	1045.812	324.632
7	Fallimento	1	1000.031	1068.719	1007.0094
15	Successo	96.6	358.688	1357.406	429.6927
15	Fallimento	3.4	1000.094	1223.125	1165.1839
23	Successo	90.8	441.25	1656.344	594.397
23	Fallimento	9.2	1000.031	1578.156	1383.9939
31	Successo	92.1	523.656	1955.125	656.1988
31	Fallimento	7.9	/	/	/

Tabella C.8: *Analisi dei principali Log di Trasmissione con inverter nella rete, polling*

Bibliografia

[1] S. Galli, A, Scaglione, Z. Wang, *For the grid and trough the Grid: The Role of Powerline Communications in the Smart Grid*, *Capitolo 2*, 2011, reperibile all'indirizzo `http://arxiv.org/abs/1010.1973`.

[2] Domologic Home Automation GmbH, *Konnex PL132 - Power-Line-Communication using the CENELEC-C-Band*, 2003, reperibile all'indirizzo `http://www.domologic.de/download/papers/index_en.html`.

[3] Aderemi A. Atayero, Adeyemi A. Alatishe, and Yury A. Ivanov, *Power Line Communication Technologies: Modeling and Simulation of PRIME Physical Layer*, *Capitolo 2*, 2012, reperibile all'indirizzo `http://www.iaeng.org/publication/WCECS2012/WCECS2012_pp931-936.pdf`.

[4] M. Rocchi, *Sistemi di comunicazione a onde convogliate: Studio e analisi comparativa*, 2012, reperibile all'indirizzo `http://amslaurea.unibo.it/4331/`.

[5] Cypress Semiconductor Corporation, *CY8CPLC20, Powerline Communication Solution*, 2014, reperibile all'indirizzo `http://www.cypress.com/documentation/datasheets/cy8cplc20-powerline-communication-solution`.

[6] Cypress Semiconductor Corporation, *User Module Datasheet: Powerline Transceiver Datasheet PLT V 1.50* , 2013, reperibile all'indirizzo `http://www.cypress.com/documentation/user-module-datasheets/user-module-datasheet-powerline-transceiver-datasheet-plt-v-150`.

[7] Cypress Semiconductor Corporation, *AN54416 - Using CY8CPLC20 in Powerline Communication (PLC) Applications* , 2015, reperibile all'indirizzo `http://www.cypress.com/documentation/application-notes/an54416-using-cy8cplc20-powerline-communication-plc-applications`.

[8] Cypress Semiconductor Corporation, *8-Bit UART Datasheet UART V 5.3*, 2015, reperibile all'indirizzo `http://www.cypress.com/documentation/user-module-datasheets/user-module-datasheet-8-bit-uart-datasheet-uart-v-53`.

[9] Cypress Semiconductor Corporation, *8-Bit Pulse Width Modulator Datasheet* , 2014, reperibile all'indirizzo `http://www.cypress.com/documentation/user-module-datasheets/user-module-datasheet-8-bit-software-pulse-width-modulator`.