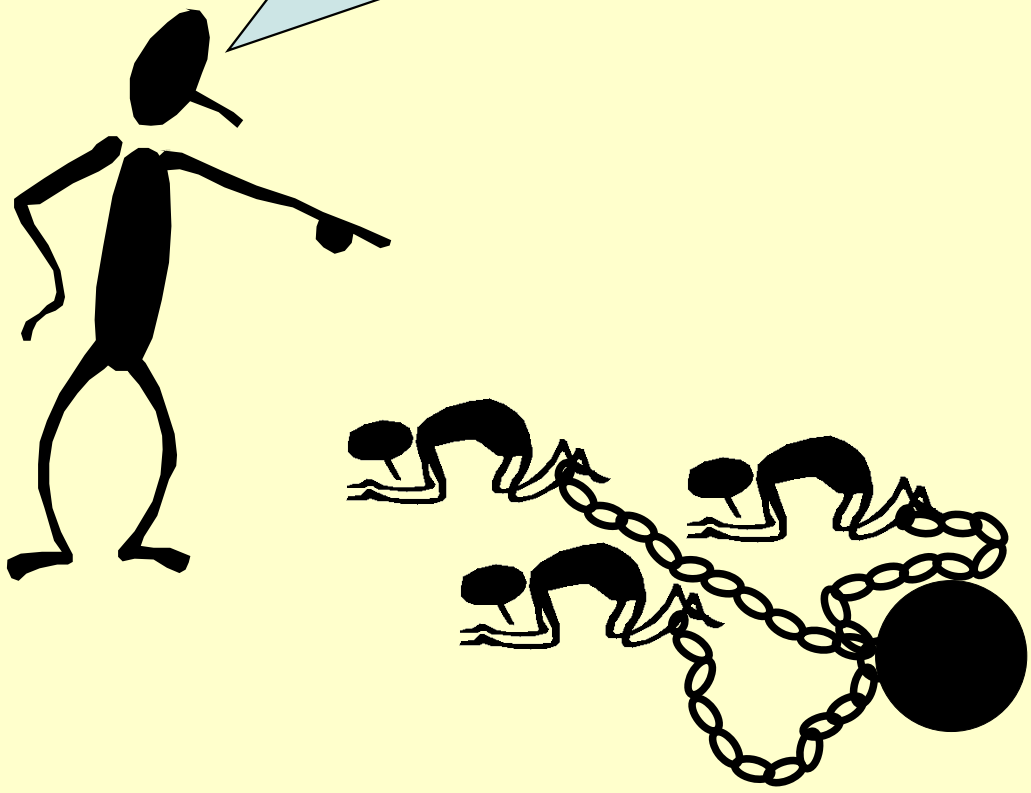
Progetto di "Laboratorio di Reti"

IL PROTOCOLLO MODBUS!



Docente:
Prof. Giovanni Rapacioli

Studente:
Mattia Cavenaghi
Matricola 736856

Anno accademico 2007/08

INDICE

CAPITOLO 1 - IL PROTOCOLLO MODBUS SERIALE

1. Introduzione

La famiglia di protocolli Modbus fu rilasciata nel 1979 dalla Schneider Automation Inc. per la creazione di reti industriali che utilizzassero i PLC Modicon, diventando in breve tempo uno *standard de facto* nei protocolli industriali di comunicazione. Oggi giorno questa famiglia di protocolli viene ampiamente utilizzata per connettere dispositivi elettronici industriali poiché:

- è open source e libera da royalty da parte della casa produttrice;
- le reti che sfruttano tale famiglia di protocolli sono facili da implementare;
- è possibile manipolare bit e parole senza porre particolari vincoli ai costruttori dei dispositivi.

Modbus è un protocollo di comunicazione a livello applicativo, posizionato al livello 7 del modello ISO/OSI che definendo la formattazione dei messaggi (*framing*), e la modalità di trasmissione dei dati e delle funzioni di controllo tra i dispositivi che comunicano secondo il paradigma client – server attraverso differenti tipi di bus o su reti eterogenee.

Il protocollo Modbus definisce un *Protocol Data Unit* (*PDU*) indipendente dal sottostante strato di comunicazione, introducendo su specifici bus e sulle reti alcuni campi addizionali definiti nella *Application Data Unit* (*ADU*) (figura 1). Dispositivi come PLC (Programmable Logic Controller), HMI (Human Machine Interface), pannelli di controllo, driver, controllori di movimento, dispositivi di I/O, etc. possono utilizzare Modbus per avviare una operazione remota e spesso il protocollo viene usato per connettere un computer supervisore con un terminale remoto (*Remote Terminal Unit*) in un sistema di supervisione, controllo ed acquisizione dei dati (SCADA).

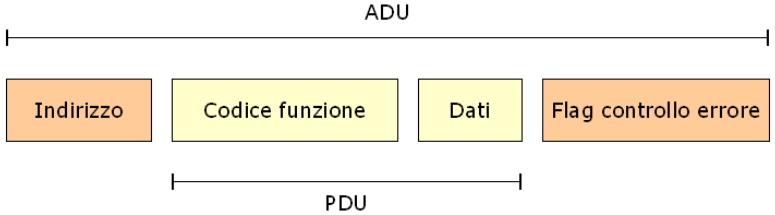


Figura 1 - Schematizzazione del framing di un messaggio secondo le specifiche del protocollo Modbus.

Le versioni ASCII ed RTU di Modbus operano su reti implementate tramite tecnologia RS 232, RS 422 e RS 485, mentre la versione del *protocollo Modbus TCP/IP* opera in particolare su tutte quelle reti che supportano la suite di protocolli TCP/IP. Utilizzando un normale gateway è possibile mettere in comunicazione diversi tipi di bus o reti (figura 2), esistono numerosi modem e gateway che supportano il protocollo Modbus poiché molto semplice da implementare e per questo spesso copiato. In alcuni casi i dispositivi vengono appositamente progettati per funzionare attraverso Modbus. La versione RTU del protocollo viene inoltre utilizzata anche nella comunicazione che sfrutta la tecnologia wireless: per questa ragione trova largo utilizzo in applicazioni coinvolte in scenari dove sono presenti sostanze tossiche o pericolose. Oltre alla tecnologia wireless e cablata, Modbus può sfruttare anche le comunicazioni di tipo SMS e GPRS. Infine Modbus viene adottato anche nei trasporti e nelle applicazioni per la produzione di energia.

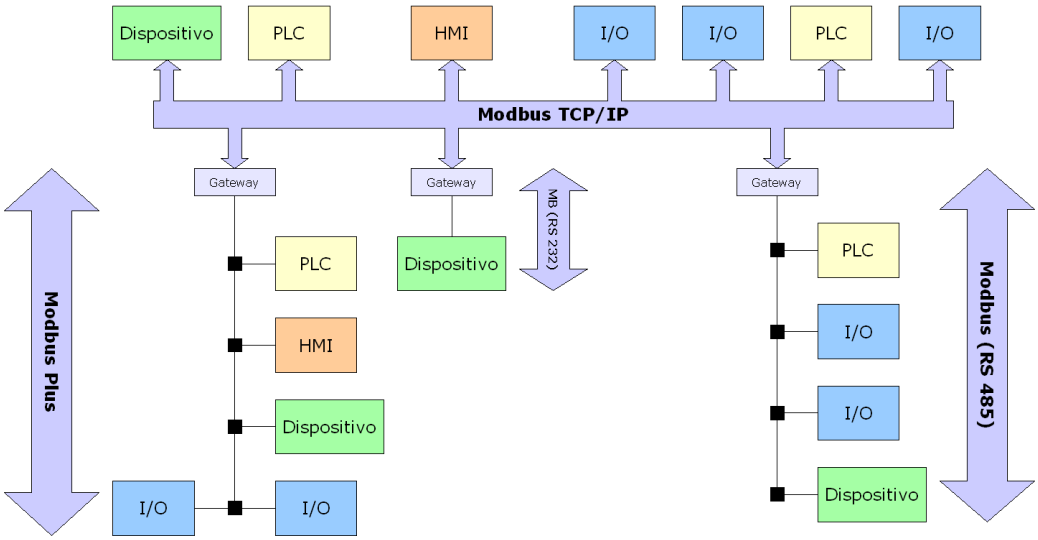


Figura 2 - Schematizzazione di una rete che sfrutta tre versioni del protocollo Modbus: Modbus seriale, Modbus TCP/IP e Modbus Plus.

2. I dispositivi master e slave

Il protocollo Modbus definisce la struttura dei messaggi spediti che i dispositivi distinti in *master* (il *client*) e *slave* (il *server*), interpretano indipendentemente dalla tipologia di rete basata sul *paradigma client – server* su cui operano. I dispositivi tipici master includono un processore host e dei pannelli di programmazione, mentre uno slave include dei controller programmabili. Ogni controller inserito nella comunicazione conosce il proprio indirizzo numerico univoco che cade nell'intervallo di valori numerici decimali 1 – 247, interpreta i messaggi ricevuti, sa distinguere il tipo di operazione da intraprendere ed estrarre le informazioni contenute nei messaggi.

Alcune funzioni di Modbus supportano la trasmissione broadcast, consentendo al master di spedire un messaggio a tutti i dispositivi slave attraverso un nodo fittizio identificato tramite l'indirizzo 0. Essendoci la possibilità di perdita di un messaggio durante il percorso, tale modalità viene sfruttata per operazioni non critiche come ad esempio la sincronizzazione temporale dei dispositivi in una rete.

Nelle reti che utilizzano la versione seriale e la versione Modbus Plus, solo il master può iniziare la comunicazione, mentre nella versione Modbus TCP/IP ogni dispositivo funge sia da master che da slave. Le tipiche operazioni che un master attraverso le funzioni

specificate per mezzo di opportuni codici funzione, può comandare ad uno slave (o più slave nella comunicazione broadcast) riguardano la modifica di un valore presente in un registro, il controllo o la lettura di una porta I/O o l'invio di uno o più valori memorizzati in un registro.

3. Il modello a registro

Nel protocollo Modbus il *registro* è la più piccola entità dotata di indirizzo e le funzioni sono definite in modo tale da operare su queste unità.

Il modello a registro è basato su una serie di tabelle con caratteristiche ben precise (tabella 1).

TABELLA	TERMINOLOGIA CLASSICA	CARATTERISTICHE
Discrete output	Coils	Dimensione 16 bit, modificabile dal programma applicativo, modalità read – write.
Discrete input	Input	Dimensione 1 bit, fornisce un sistema di I/O, modalità read – only.
Input register	Input register	Dimensione 16 bit, fornisce un sistema di I/O, modalità read – only.
Output register	Holding register	Dimensione 1 bit, modificabile dal programma applicativo, modalità read – write.

Tabella 1 - Tipologie di registro e loro caratteristiche, presenti nelle specifiche del protocollo Modbus.

Il protocollo Modbus seriale definisce due tipi elementari di dato: il *tipo discreto* ed il *tipo registro*. Il tipo di dato discreto rappresenta un valore bit usato per indirizzare gli output coils ed i digital input di un PLC, mentre un tipo di dato registro rappresenta valori interi a 16 bit, è prassi comune trasferire valori float ed interi a 32 bit come una coppia di due valori di tipo registro a 16 bit in *ordine little endian* (i byte meno significativi di un valore vengono memorizzati prima dei byte più significativi).

Questi tipi di dato, ad esclusione dei valori a 32 bit, vengono trasferiti secondo l'*ordine big – endian* ossia i byte più significativi vengono memorizzati prima dei byte meno significativi.

Il protocollo Modbus definisce queste aree in modo libero: la distinzione tra input ed output, indirizzamento a bit ed indirizzamento a registro dei dati non implica particolari comportamenti da parte dello slave, è tipico però che i dispositivi slave, per questioni di efficienza implementino i costrutti logici riportati nella tabella 1 come aree di memoria sovrapposte.

Per ogni tabella il protocollo consente l'accesso di un massimo di 65536 oggetti, ed è lo slave a stabilire se tali oggetti siano o meno accessibili dal master. Tipicamente lo slave consente l'accesso di solo una piccola area di memoria come ad esempio 1024 byte.

4. Versioni del protocollo Modbus seriale

Il protocollo Modbus seriale fornisce due differenti modalità di formattazione dei

messaggi, a seconda del tipo di applicazione che si vuole implementare. *Modbus RTU* fornisce una rappresentazione binaria e quindi compatta dei dati, mentre *Modbus ASCII* è una versione più "umana" poiché i dati (indirizzi, codici funzione, etc.) sono leggibili da parte dell'utente essendo codificati in formato esadecimale.

Entrambe le versioni utilizzano la comunicazione di tipo seriale, nel formato RTU ai comandi ed ai dati viene fatto seguire un *checksum CRC* (*Cyclic Redundancy Checksum*), mentre nel formato ASCII si genera un *checksum LRC* (*Longitudinal Redundancy Checksum*). I nodi configurati per la variante RTU non possono comunicare con i nodi ASCII e viceversa.

Come vedremo nel seguito di questo capitolo, i tipi di dato e le chiamata a funzione sono identiche per entrambe le varianti del protocollo di comunicazione: solo la formattazione dei messaggi risulta differente.

4.1.La versione ASCII

La versione ASCII del protocollo usa una codifica ASCII dei dati e un checksum di 8 bit. Il frame del messaggio è delimitato dal carattere ":" all'inizio e dalla sequenza "Carriage Return – Line Feed" (CRLF) alla fine dello stesso.

La messaggistica ASCII è meno efficiente e sicura della messaggistica RTU ed andrebbe impiegata solamente con dispositivi che non supportano il formato RTU. Un altro utilizzo della versione ASCII di Modbus è nelle reti di comunicazione dove l'RTU non è applicabile poiché i dati non possono essere trasmessi come un flusso continuo.

La messaggistica ASCII è *state – less* non è necessario quindi aprire e conseguentemente chiudere una connessione ad un dispositivo slave od effettuare particolari procedure di ripristino dai possibili errori di trasmissione.

4.2.La versione RTU

Il protocollo RTU utilizza la codifica binaria dei dati ed un CRC a 16 bit per il rilevamento degli errori di trasmissione. I frame del messaggio sono delimitati da un intervallo di silenzio pari alla trasmissione di 3.5 caratteri prima e dopo il messaggio.

Quando si utilizza il protocollo RTU è molto importante che il messaggio sia spedito in modo continuo senza gap, in caso contrario ossia al verificarsi di un gap di tempo maggiore all'invio di 3.5 caratteri, lo slave interpreta ciò come la fine del messaggio e scarta i bit ricevuti in seguito.

Anche in tal caso la messaggistica RTU è senza stato e non è necessario aprire o chiudere eventuali connessioni al dispositivo slave, o ricorrere a delle procedure di recovery dagli errori.

5. Limitazioni del protocollo Modbus seriale

Poiché Modbus è un protocollo che sfrutta il paradigma master – slave non c'è modo per un dispositivo di forzare una eccezione, il nodo master deve continuamente interrogare ogni campo del dispositivo slave per la ricerca di eventuali modifiche nei dati. Questo consuma banda e tempo di connessione in applicazioni in cui la banda può essere costosa come in un collegamento radio.

L'indirizzamento Modbus permette il collegamento dati tra un massimo di 254 dispositivi, fattore che limita il numero di dispositivi che possono essere connessi ad una stazione master.

6. Transazioni sulle reti Modbus seriale

I dispositivi comunicano usando una tecnica master – slave in cui un dispositivo chiamato master può iniziare una transazione mediante l'invio di un messaggio chiamato *query*. I dispositivi slave rispondono attraverso un messaggio chiamato *response*, fornendo al master i dati od eseguendo le azioni richieste.

Come abbiamo visto in precedenza il master può interrogare slave individuali od indirizzare un messaggio in modalità broadcast, le response vengono generate solamente nel caso di interrogazioni singole e non broadcast.

Il protocollo Modbus stabilisce il formato delle query del master immettendo sul bus di comunicazione l'*indirizzo* del dispositivo destinatario, un *codice funzione* che definisce l'azione da intraprendere, i *dati* da spedire, ed un *campo di controllo degli errori*.

La response contiene un campo con il codice di riferimento dell'azione eseguita, i dati da ritornare al master ed un campo di controllo degli errori. Se si verificasse un errore nella ricezione del messaggio, o se lo slave non fosse in grado di effettuare l'azione richiesta, il messaggio di response fornisce al master un codice per l'identificazione dell'errore verificatosi.

Tenendo presente la struttura dei messaggi trasmissibili sulle reti Modbus seriale (figura 1) possiamo distinguere tali costrutti in:

- *Query:* contenente un codice funzione che comanda al dispositivo slave quale azione intraprendere. I byte nel campo dati contengono qualsiasi informazione addizionale che consente allo slave di eseguire la funzione specificata, l'indirizzo di inizio ed il numero di registri da leggere. Il campo di controllo degli errori fornisce un meccanismo di validazione dell'integrità del contenuto del messaggio ricevuto.
- *Response:* in caso di assenza di errori di ricezione, lo slave inserisce nella sua risposta il codice funzione contenuto nella query. I byte nel campo dati contengono i dati collezionati dallo slave, i quali riportano i valori dei registri richiesti dal master. Nel caso si verificasse un errore il codice funzione viene modificato in modo tale da far rilevare al dispositivo master che la response è un messaggio di errore e nel campo dati viene inserita la descrizione dello stesso. Anche in questo caso il campo di controllo dell'errore permette al master di verificare la validità del messaggio ricevuto dallo slave.

7. Due metodi di trasmissione seriale

Abbiamo accennato che i dispositivi possono essere impostati per comunicare su una rete Modbus standard usando due modalità di trasmissione, *ASCII* od *RTU*. L'utente durante la configurazione di ogni dispositivo seleziona la modalità attraverso una serie di parametri relativi alle porte di comunicazione seriale (come ad esempio il rate di baud). La modalità ed i parametri della comunicazione seriale devono essere gli stessi per tutti i dispositivi posti sulla medesima rete Modbus.

La selezione del modo ASCII od RTU riguarda solo lo standard delle reti Modbus, esso definisce come i bit contenuti nei campi nel messaggio vengono trasmessi e determina come le informazioni vengono impacchettate nei singoli campi del messaggio per poi essere successivamente decodificati.

7.1.La versione ASCII

Quando i dispositivi sono impostati per comunicare su una rete Modbus che utilizza la modalità ASCII, ogni byte in un messaggio è spedito come due caratteri ASCII. Il vantaggio principale di questo metodo è che permette di avere delle interruzioni tra caratteri fino ad un secondo di tempo, senza che si generino errori.

Le peculiarità della modalità ASCII sono:

- *Sistema di codifica:* esadecimale (0 – 9, A – F);
- *Bit per byte:* 1 bit di inizio, 7 bit di dati, 1 bit per parità – disparità, 1 bit di fine messaggio se viene attivato il controllo di parità, altrimenti 2 bit;
- *Campo di controllo degli errori:* codice LRC.

7.2.La versione RTU

Quando i dispositivi sono impostati per comunicare su reti Modbus che utilizzano la modalità RTU, ogni byte in un messaggio contiene due caratteri esadecimali. Il principale vantaggio di questa modalità è che a parità di rate di baud, vi è una maggiore densità di dati nella spedizione, rispetto alla modalità ASCII ed ogni messaggio deve essere spedito mediante un flusso continuo.

Le peculiarità della modalità RTU sono:

- *Sistema di codifica:* valori binari ad 8 bit corrispondenti a due caratteri esadecimali per ogni campo del messaggio;
- *Bit per byte:* 8 bit di dati, 1 bit per la parità – disparità, 1 bit di fine messaggio se viene attivato il controllo di parità, altrimenti 2 bit;
- *Campo di controllo degli errori:* codice CRC.

8. Framing dei messaggi Modbus

In entrambe le due modalità di trasmissione i messaggi vengono inserti in un frame che ha un punto d'inizio ed una fine ben precisi. Questo permette al dispositivo ricevente, leggendo il messaggio dall'inizio, di leggere la porzione relativa all'indirizzo, e determinare a quale dispositivo è diretto il messaggio e sapere quando il messaggio termina. I messaggi incompleti possono essere cancellati e gli errori ritornati nel messaggio di response.

8.1.La versione ASCII

Nella modalità ASCII il messaggio comincia con il carattere ":" che in formato esadecimale corrisponde al valore "3A" e termina con i caratteri "carriage return – line feed" in formato esadecimale rispettivamente "0D" e "0A".

Tutti i caratteri trasmissibili per i campi sono esadecimali, i dispositivi slave

monitorizzano il bus in modo continuo alla ricerca del carattere ":" e quando ne ricevono uno, viene decodificato il campo successivo (il campo indirizzo) per scoprire qual'è il dispositivo ricevente.

Tra i caratteri di un messaggio possono verificarsi degli intervalli di silenzio fino ad un secondo di tempo. Se si verificano intervalli più lunghi il dispositivo ricevente presume che si sia verificato un errore.

Nella seguente tabella 2 è riportato il frame di un tipico messaggio ASCII.

INIZIO	INDIRIZZO	FUNZIONE	DATI	CONTROLLO LRC	FINE
1 carattere	2 caratteri	2 caratteri	N caratteri	2 caratteri	2 caratteri CRLF

Tabella 2 - Framing di un messaggio secondo la modalità ASCII.

8.2.La versione RTU

Nella modalità RTU il messaggio inizia con un intervallo di tempo di silenzio pari all'invio di almeno 3.5 caratteri. Questa caratteristica viene spesso implementata attuando un intervallo di tempo pari al multiplo dell'invio di un numero di caratteri uguale al rate di baud usato nella rete (T1 T2 T3 T4 nella tabella 3).

I caratteri disponibili per ogni campo sono in formato binario, i dispositivi di rete controllano in modo ciclico il bus di rete inclusi gli intervalli di silenzio. Quando il primo campo di indirizzo viene ricevuto, ogni dispositivo decodifica se è il dispositivo destinatario del messaggio.

Trasmesso l'ultimo carattere del messaggio segue un intervallo di tempo pari all'invio di 3.5 caratteri che identifica la fine del messaggio. Dopo questo intervallo può essere spedito un nuovo messaggio.

L'intero messaggio deve essere trasmesso come un flusso continuo di dati. Se un intervallo di tempo di silenzio di 1.5 caratteri avviene prima del completamento del frame, il dispositivo ricevente elimina il messaggio incompleto ed assume che il prossimo byte ricevuto sia il campo indirizzo di un nuovo messaggio.

Se un messaggio inizia prima dell'intervallo di tempo pari all'invio di 3.5 caratteri, il dispositivo ricevente assume che i dati ricevuti siano la continuazione del precedente messaggio. Questo genera un errore in modo tale che il campo CRC non sia valido per i messaggi originali e ricevuti. Il frame di un tipico messaggio RTU è riportato nella seguente tabella 3.

INIZIO	INDIRIZZO	FUNZIONE	DATI	CRC	FINE
T1 T2 T3 T4	8 bit	8 bit	N × 8 bit	16 bit	T1 T2 T3 T4

Tabella 3 - Framing di un messaggio secondo la modalità RTU.

9. Il campo indirizzo del messaggio

Il campo indirizzo di un messaggio contiene due caratteri (nella modalità ASCII) od otto bit (nella modalità RTU). Gli indirizzi validi assegnabili ai dispositivi slave vanno dal valore numerico decimale 1 al valore 247 mentre il valore numerico 0 riconosciuto da tutti gli slave, è riservato per la comunicazione broadcast.

Un dispositivo master indirizza un messaggio ad uno specifico slave, scrivendo l'indirizzo

del dispositivo destinatario nel campo indirizzo del messaggio di query. Quando lo slave risponde riscrive il proprio indirizzo nel campo indirizzo del messaggio di response, in modo tale da confermare l'avvenuta ricezione dell'interrogazione.

10.Il campo del codice funzione del messaggio

Anche in questo caso il campo funzione contiene due caratteri per la modalità ASCII od otto bit per la modalità RTU. I codici validi sono numerati dal valore decimale 1 al valore 255, di questi alcuni sono riservati ai controller Modicon o ad altri particolati modelli.

Quando un messaggio viene spedito da un master allo slave, il campo funzione specifica quale azione lo slave deve eseguire (leggere o scrivere registri, caricare o verificare il programma dello slave, etc.).

Quando, come nel caso del campo indirizzo, lo slave costruisce il messaggio di response reinserisce il codice funzione ricevuto dal dispositivo master. Nel caso in cui non si siano verificati errori di esecuzione, lo slave ripete semplicemente il codice originale, in caso contrario ritorna il codice originale col bit più significativo impostato ad 1. Ad esempio se il codice funzione ricevuto dal dispositivo master corrisponde al valore binario 0000 0011 (03 in formato esadecimale), lo slave in caso di errore spedirà al master il codice binario 1000 0011 (83 in formato esadecimale).

Al verificarsi di un evento di eccezione lo slave imposta un codice numerico univoco nel campo dati della response, questo comunica al master che tipo di errore si è verificato o la ragione dell'eccezione.

Alcuni tipici codici funzione con le rispettive descrizioni sono riportati nella seguente tabella 4.

CODICE	NOME	DESCRIZIONE
01	Read Coil Status	Legge il valore dei discrete output di uno slave.
02	Read Input Status	Legge il valore dei discrete input di uno slave.
03	Read Holding Register	Legge il valore binario degli holding register di uno slave.
04	Read Input Register	Legge il valore binario degli input register di uno slave.
05	Force Single Coil	Forza un coil al valore ON od OFF.
...		
08	Diagnostics	Esegue le funzionalità di diagnostica, per i codici di sub funzione osservare la tabella 5.
...		

Tabella 4 - Esempi di codici funzione del protocollo Modbus seriale.

10.1.Le tipologie di funzione

Le funzioni eseguibile dagli slave che ricevono i messaggi costruiti secondo il protocollo Modbus, si suddividono in tre classi:

- *Public Function Codes* (codici funzione pubblici): sono funzioni uniche, garantite e

documentate dalla comunità Modbus-IDA.org attraverso un apposito processo di validazione.

- *User – Defined Function Codes* (codici funzione definiti dall'utente): ci sono due range di codici definiti dall'utente che spaziano dal valore numerico 65 al valore numerico 72 e dal valore numerico 100 al valore 110. Un utente può scegliere di implementare una funzione non implementata nelle specifiche del protocollo, ma non vi è garanzia che l'utilizzo di tale funzione sia unico. Se l'utente volesse rendere pubblica tale funzione è necessario che quest'ultima venga sottoposta all'analisi per RFC da parte della Modbus Organization, la quale si riserva il diritto di sviluppare la RFC proposta.
- *Reserved Function Codes* (codici funzione riservati): sono codici utilizzati da alcune compagnie dedicati a prodotti proprietari e quindi non accessibili per il pubblico utilizzo.

10.2.Le funzioni di diagnostica

Il protocollo Modbus seriale fornisce inoltre una serie di test per il controllo della comunicazione tra il master e lo slave o per il controllo di varie condizioni di errore verificate all'interno dello slave, per le funzioni di diagnostica la modalità di spedizione broadcast non è supportata.

Per eseguire la funzionalità di diagnostica il master oltre al codice funzione, inserisce nella query un codice di sub funzione composto da due byte, definendo così il tipo di test da effettuare: lo slave nel caso di normale funzionamento risponde ripetendo entrambi i codici di funzione e di sub funzione.

Molte delle query di diagnostica utilizzano due byte inseriti nel campo dati, per spedire dati di diagnostica od informazioni di controllo. Alcune delle operazioni di diagnostica consistono semplicemente nel rispedire al master i dati contenuti nel campo dati dello slave.

Eseguire una funzione di diagnostica sullo slave non compromette il normale funzionamento del programma utente. La logica del programma così come i valori discreti ed i valori di registro non vengono modificati dalla funzione, ma certe funzioni possono richiedere semplicemente di reinizializzare il campo contatore degli errori nel dispositivo slave.

Un dispositivo slave può in ogni modo essere forzato alla modalità di solo ascolto in cui monitorizza i messaggi della comunicazione di sistema senza rispondere. Questo può influire sul risultato del programma utente ma ciò dipende dal tipo di comunicazione che si è scelto di implementare. La modalità di solo ascolto rimuove i dispositivi malfunzionanti dalla comunicazione di sistema.

Nella seguente tabella 5 sono riportati a scopo di esempio alcuni codici (sub funzione) di diagnostica.

CODICE (SUB FUNZIONE)	NOME	DESCRIZIONE
00	Return Query Data	Vengono rispediti al dispositivo master tramite una response, i dati contenuti nel campo dati della query.
...		
02	Return Diagnostic Register	Tramite una response viene spedito al dispositivo master il contenuto del registro di diagnostica a 16 bit del dispositivo slave.
...		
04	Force Listen Only Mode	Forza il dispositivo slave alla modalità di solo ascolto.
05 – 09	Riservati	
10	Clear Cts and Diagnostic Reg.	Pulisce i registri contatore e di diagnostica del dispositivo slave.
...		

Tabella 5 - Esempi di codici di sub funzione per le operazioni di diagnostica.

11.Il campo dati del messaggio

Il campo dati è costruito usando coppie di cifre esadecimali che ricadono nell'intervallo 00 – FF, questo campo può essere implementato tramite due caratteri ASCII od un carattere binario da otto bit nella modalità RTU, a seconda del tipo di trasmissione seriale impostato per la rete. In certi tipi di messaggio il campo dato può addirittura essere vuoto di lunghezza 0.

Il campo dati di un messaggio spedito dal master allo slave può contenere informazioni addizionali che servono allo slave per eseguire l'azione richiesta dal campo funzione, questo può riguardare l'indirizzo dei registri, valori discreti, la quantità di oggetti da manipolare, od il numero di byte dati nel campo.

Se non si verificano errori il campo dati di una response spedita dallo slave contiene i dati richiesti dal master. Al verificarsi di un errore il campo contiene un codice di eccezione che il programma mater utilizza per determinare la prossima azione da intraprendere.

12.Metodi di controllo degli errori

Le reti che utilizzano il protocollo Modbus seriale contemplano due tipologie di campo di controllo degli errori. Il campo del controllo della parità (per la parità o disparità) che può essere opzionalmente applicato ad ogni carattere ed il campo per il controllo del frame del messaggio effettuato mediante il calcolo dell'LRC o del CRC a seconda della versione di Modbus utilizzata. Entrambi i metodi di controllo sono generati dal dispositivo master ed applicati al contenuto del messaggio prima dell'invio allo slave. Il dispositivo di slave controlla ogni carattere e l'intero frame del messaggio durante la ricezione.

Il master viene configurato dall'utente in modo da rimanere in attesa per un determinato periodo di tempo predeterminato prima di abortire la transazione. Tale intervallo è impostato per essere lungo abbastanza in modo da permettere allo slave di rispondere: se lo slave rileva un errore di trasmissione, il messaggio non viene accettato e non viene

costruito il messaggio di response, facendo scadere il tempo di timeout e consentendo al programma master di far fronte all'errore generato. Nel caso in cui il master abbia generato ed inviato un messaggio destinato ad un dispositivo slave inesistente, l'intervallo di tempo fornito dal master scade, generando così un errore di trasmissione.

12.1.Il controllo della parità

L'utente può configurare i dispositivi per il controllo della parità, disparità o per la "non parità", questo determina come il bit di controllo della parità viene impostato per ogni carattere inviato.

L'attivazione del controllo di parità o disparità comporta il conteggio dei bit impostati al valore binario 1 per ogni carattere contenuto nel campo dati, il bit di parità viene impostato a 0 se il numero dei bit è pari od 1 se è dispari.

Quando il messaggio viene trasmesso il bit di parità viene calcolato sul frame di ogni carattere, il dispositivo di ricezione conta la quantità di bit impostati al valore binario 1 e nel caso in cui il bit di parità o disparità non sia uguale a quanto impostato per l'intera rete si genera un errore (tutti i dispositivi sulla rete Modbus devono essere configurati per usare lo stesso metodo di controllo della parità).

Se non viene attivato il controllo della parità, non viene trasmesso nessun bit atto a questa funzionalità ed al suo posto viene inserito un bit aggiuntivo di fine messaggio, per il riempimento del frame.

12.2.La versione ASCII: il calcolo del campo LRC

Nella modalità ASCII il messaggio include un campo di controllo degli errori basato sul *metodo LRC (Longitudinal Redundancy Check)*. Il campo LRC controlla il contenuto del messaggio ad esclusione del valore ":" e del "CRLF" e viene applicato indipendentemente dal metodo di controllo della parità utilizzato per ogni carattere del messaggio.

Il campo LRC consiste in un byte calcolato dal dispositivo che spedisce il messaggio ed appeso alla fine dello stesso. Il dispositivo ricevente calcola LRC durante la ricezione del messaggio e compara il valore calcolato col valore ricevuto, nel caso in cui i due valori fossero discordi viene generato un errore.

Il valore LRC viene calcolato facendo la somma dei byte contenuti nel messaggio (ad esclusione dei caratteri ":" e "CRLF"), scartando il riporto ed effettuando l'operazione di complemento a 2 del risultato. Essendo il risultato la somma progressiva di valori ad otto bit, si ottiene un valore maggiore di 255, comportando così la reinizializzazione del campo al valore decimale 0. Non essendoci un nono bit per la memorizzazione del riporto, questo viene scartato automaticamente.

La procedura di generazione del codice LRC è riassumibile nei seguenti tre punti:

1. Somma di tutti i byte del messaggio ad esclusione dei caratteri ":" e "CRLF". Inserimento del valore ottenuto in un campo ad otto bit, scartando il riporto.
2. Sottrazione al valore di otto bit ottenuto come risultato della precedente operazione, del valore esadecimale FF (in formato binario corrisponde ad un valore di 1111 1111) in modo tale da riprodurre l'operazione di complemento ad uno.

3. Somma del valore decimale 1 al risultato ottenuto al precedente punto.

Il valore ottenuto dal procedimento eseguito nei tre punti enunciati corrisponde al codice LRC del messaggio che viene appeso in coda allo stesso e trasmesso in modalità big – endian, se ad esempio il codice LRC del messaggio corrisponde al valore esadecimale 61, il frame corrisponderà a quanto riportato nella seguente tabella 6.

:	Indirizzo	Codice funzione	Contatore dei dati	Dati	Dati	Dati	Dati	***LRC Hi: 6***	***LRC Lo: 1***	CR	LF

Tabella 6 - Esempio di frame di un messaggio ASCII: memorizzazione del campo LRC.

12.3.La versione RTU: il calcolo del campo CRC

Nella modalità RTU il messaggio contiene un campo di controllo errori basato sul *metodo CRC (Cyclic Redundancy Check)*. Come nel caso del calcolo del campo LRC della modalità ASCII, il campo CRC viene calcolato sull'intero contenuto del messaggio ed indipendentemente dal metodo di controllo della parità adottato.

Il campo CRC è composto da due byte contenenti un valore binario a 16 bit. Il valore CRC è calcolato dal dispositivo trasmittente che appende il valore alla fine del messaggio, ed il dispositivo ricevente lo ricalcola durante la ricezione del messaggio. Se i due valori (valore calcolato e valore ricevuto) sono discordi il dispositivo ricevente genera un messaggio di errore.

Il processo di generazione del CRC si può riassumere in sei fasi ben distinte:

1. Caricamento in un registro di un valore binario a 16 bit formato da valori binari 1, e chiamato *registro di CRC*.
2. Esecuzione dello XOR dei primi otto bit di un messaggio con i byte di ordine inferiore del registro di CRC, ed inserimento del risultato nel registro di CRC.
3. Esecuzione dello Shift di un bit a destra del registro di CRC, impostazione al valore binario zero del MSB (Most Significant Bit), estrazione ed esame dell'LSB (Least Significant Bit).
4. Se il bit LSB è pari al valore binario 0 viene ripetuto il passo numero 3, altrimenti si esegue lo XOR del registro CRC con il valore "A001" esadecimale.
5. Ripetizione del passo 3 e 4 finché non vengono eseguite 8 operazioni di Shift, concludendo così il processo del primo byte del messaggio.
6. Ripetizione delle operazioni dal passo 2 al passo 5 per i prossimi byte del messaggio, iterando il procedimento finché tutti i byte sono stati processati.

Il valore contenuto nel registro di CRC è il campo CRC ricercato e viene memorizzato nel frame del messaggio inserendo dapprima il byte di ordine superiore seguito dal byte di ordine inferiore (modalità little – endian). Se ad esempio il codice CRC è pari al valore esadecimale 1241 (in formato binario 0001 0010 0100 0001), nel messaggio viene memorizzato tramite la formattazione riportata nella seguente tabella 7.

Indirizzo	Codice funzione	Contatore dei dati	Dati	Dati	Dati	Dati	**CRC Hi: 41**	**CRC Lo: 12**

Tabella 7 - Esempio di frame di un messaggio RTU: memorizzazione del campo CRC.

13.Esempio di trasmissione di una query ed una response

Concludendo questo primo capitolo dedicato al protocollo Modbus seriale, è possibile riassumere la sequenza di eventi che si verificano all'atto di una transazione Modbus, mediante la seguente figura 3.

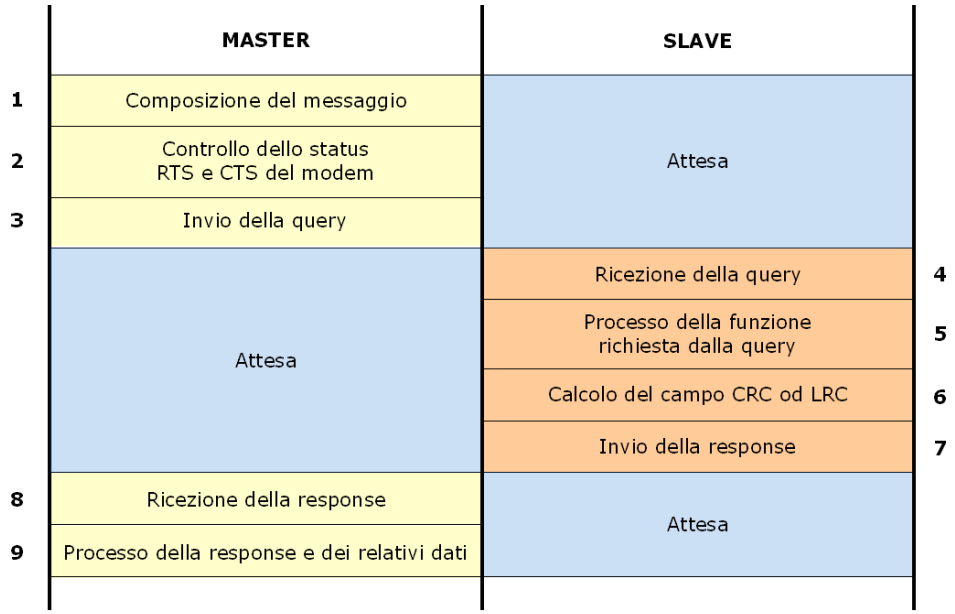


Figura 3 - Diagramma delle sequenze di eventi che comporta l'esecuzione di una transazione Modbus.

Dopo la composizione del messaggio si effettua la scansione dello status dei pin RTS e CTS del modem (Request To Send e Clear To Send, pin per il controllo del flusso dei dati inviati sulla linea RS232), successivamente la query viene inviata al dispositivo slave il cui tempo di spedizione è calcolabile mediante la seguente formula:

$$\text{Tempo (ms)} = \frac{1000 \times \text{numero di caratteri} \times \text{bit per carattere}}{\text{rate di baud}} .$$

Il messaggio Modbus viene processato al temine del controllo da parte del dispositivo slave. Nel caso peggiore il tempo richiesto per tale operazione è pari ad una scansione dell'intero messaggio. Il tempo medio di scansione è invece pari alla scansione di metà messaggio.

Per il calcolo del campo LRC è invece richiesto meno di 1 ms mentre per il campo CRC sono richiesti 0.3 ms per ogni 8 bit di dati ritornati nella response.

Le tabelle 8 e 9 mostrano due messaggi uno di query ed uno di response oggetto di un'ipotetica transazione. In entrambi gli esempi sono riportati i dati in formato esadecimale e la relativa traduzione nelle due modalità ASCII ed RTU. La query di esempio richiede semplicemente allo slave di leggere il contenuto di un particolare registro di holding indirizzato col valore 06, e di ritornare i dati in esso contenuti al dispositivo master.

Come è possibile osservare nella tabella 9 la risposta dello slave include la ripetizione del codice funzione comunicato dal master, indice del normale funzionamento del dispositivo oggetto della transazione. Il byte count indica il numero di oggetti byte presenti nel messaggio che devono essere ritornati.

NOME DEL CAMPO	VALORE DI ESEMPIO IN FORMATO ESADECIMALE	VALORE DI ESEMPIO TRASMESSO NELLA MODALITÀ **ASCII**	VALORE DI ESEMPIO TRASMESSO NELLA MODALITÀ **RTU**
Intestazione		*:*	
Indirizzo dello slave	06	*06*	*0000 0110*
Codice funzione	03	*03*	*0000 0011*
Indirizzo di inizio HI	00	*00*	*0000 0000*
Indirizzo di inizio LO	6B	*6B*	*0110 1011*
Numero di registri HI	00	*00*	*0000 0000*
Numero di registri LO	03	*03*	*0000 0011*
Campo di controllo dell'errore		*LRC a 2 caratteri*	*CRC a 16 bit*
Trailer		*CRLF*	

Tabella 8 - Esempio di messaggio query spedito dal dispositivo master.

NOME DEL CAMPO	VALORE DI ESEMPIO IN FORMATO ESADECIMALE	VALORE DI ESEMPIO TRASMESSO NELLA MODALITÀ **ASCII**	VALORE DI ESEMPIO TRASMESSO NELLA MODALITÀ **RTU**
Intestazione		*:*	
Indirizzo dello slave	06	*06*	*0000 0110*
Codice funzione	03	*03*	*0000 0011*
Conteggio byte	06	*06*	*0000 0110*
Dati HI	02	*02*	*0000 0010*
Dati LO	2B	*2B*	*0010 1011*
Dati HI	00	*00*	*0000 0000*
Dati LO	00	*00*	*0000 0000*
Dati HI	00	*00*	*0000 0000*
Dati LO	63	*63*	*0110 0011*
Campo di controllo dell'errore		*LRC a 2 caratteri*	*CRC a 16 bit*
Trailer		*CRLF*	

Tabella 9 - Esempio di messaggio response spedito dal dispositivo slave.

CAPITOLO 2 - IL PROTOCOLLO MODBUS TCP/IP

1. Introduzione

Oltre alla versione seriale di Modbus esiste una versione dedicata alle reti che sfruttano la suite di protocolli TCP/IP. Il protocollo Modbus TCP/IP sviluppato nel 1999, è appunto una variante di Modbus seriale RTU basato sul protocollo TCP/IP che permette l'invio di messaggi su reti Intranet ed Internet.

TCP/IP è il protocollo di trasporto di internet ed attualmente è un insieme di protocolli stratificati che forniscono un meccanismo affidabile di trasporto dei dati tra macchine, in questa versione di Modbus è utilizzata soprattutto la tecnologia Ethernet, divenuta lo standard de facto per il networking industriale. Non è una tecnologia nuova, ma maturata con il progressivo decadimento dei costi di implementazione delle moderne reti di comunicazione.

Il protocollo Modbus TCP/IP è sempre più utilizzato perché è open source, semplice da implementare, ha un basso costo di sviluppo e richiede un supporto hardware minimo, inoltre esistono centinaia di dispositivi compatibili disponibili sul mercato. Modbus TCP/IP è utilizzato per scambiare informazioni tra dispositivi, programmi applicativi e programmi di monitoraggio. È inoltre utilizzato per gestire sistemi di I/O distribuiti divenendo il protocollo preferito tra le aziende che sviluppano questa classe di dispositivi.

Il protocollo Modbus TCP/IP utilizza la codifica binaria dei dati ed il meccanismo di rilevamento TCP/IP degli errori di trasmissione. A differenza alle varianti ASCII ed RTU della versione seriale, Modbus TCP/IP è orientato alle connessioni e permette di eseguirle in modo concorrente sullo stesso slave così come su più dispositivi.

Implementare i driver compatibile con Modbus TCP/IP risulta semplice poiché tale protocollo corrisponde al Modbus seriale dotato di un incapsulamento TCP. Per tale comunicazione si necessità di due dispositivi, uno master ed uno slave e possono essere due PC che dialogano attraverso schede Ethernet, od un PC che interagisce con un sensore od un dispositivo avente un microcomputer incorporato. Un dispositivo gateway è sempre necessario per convertire i dati dal livello fisico ad Ethernet, e convertire il protocollo Modbus seriale a Modbus TCP/IP, tale gateway può essere implementato utilizzando un normale PC.

2. Il paradigma client – server

Il protocollo Modbus TCP/IP utilizza anch'esso il paradigma master – slave, nella variante client – server tra dispositivi connessi ad una rete Ethernet TCP/IP. In questo tipo di comunicazione si utilizzano inoltre quattro tipi di messaggio (figura 4).

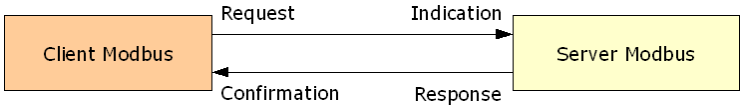


Figura 4 - Diagramma dello scambio di messaggi che si verifica in una comunicazione Modbus TCP/IP.

- *Modbus Request:* è un messaggio spedito sulla rete dal dispositivo client che permette di iniziare una transazione;
- *Modbus Confirmation:* è il messaggio di Response ricevuto dal dispositivo client;
- *Modbus Indication:* è il messaggio di Request ricevuto dal dispositivo server;
- *Modbus Response:* è il messaggio di risposta alla Request inviato dal dispositivo server.

Il servizio di scambio dei messaggi Modbus TCP/IP è utilizzato per lo scambio di dati in modalità real time tra due dispositivi applicativi, tra dispositivi applicativi ed altri dispositivi, tra applicazioni HMI (*Human Machine Interface*) o SCADA e dispositivi, e tra PC e dispositivi applicativi che forniscono servizi on – line.

Solo il dispositivo client identificato come master può iniziare una transazione, costruendo l'ADU del messaggio, il cui codice funzione indica al server quale azione intraprendere.

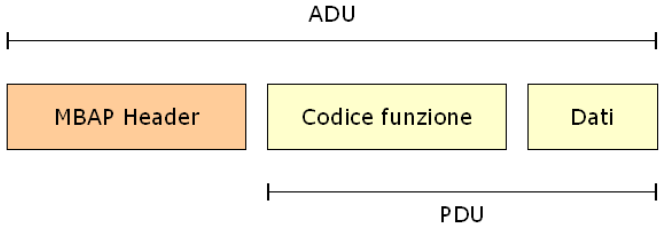


Figura 5 - Framing di un messaggio costruito utilizzando il protocollo Modbus TCP/IP.

3. L'header Modbus TCP/IP

Nel protocollo Modbus TCP/IP un header, chiamato *Modbus Application Protocol header*, è dedicato all'identificazione dell'ADU, che a differenza della versione seriale RTU è caratterizzato da un byte di *unit identifier* che corrisponde all'indirizzo dello slave nel protocollo Modbus RTU. Tale byte è utilizzato per comunicare attraverso un dispositivo atto ad operare come un bridge, un router od un gateway, utilizzando un singolo indirizzo IP per supportare più unità indipendenti.

Tutte le request e response Modbus TCP/IP vengono realizzate in modo da permettere al dispositivo ricevente di verificare il termine del messaggio. Nel caso il PDU del messaggio Modbus abbia lunghezza fissa il codice funzione può essere specificato da solo, nel caso in cui invece il codice funzione venga spedito assieme ad un numero variabile di dati, nel campo dati del messaggio viene incluso anche un contatore di byte.

L'implementazione di Modbus sul protocollo TCP/IP comporta l'inserimento di

informazioni aggiuntive nell'MBAP header in modo da permettere al dispositivo ricevente di identificare i limiti del messaggio, operazione particolarmente utile se il messaggio è stato spezzettato in più pacchetti. L'esistenza di regole per il calcolo della lunghezza e l'utilizzo del CRC – 32 su Ethernet permette di identificare in modo sicuro i messaggi danneggiati.

Le informazioni contenute nell'MBAP header sono riassumibili nella seguente tabella 10.

CAMPO	LUNGHEZZA	DESCRIZIONE	DISPOSITIVO CLIENT	DISPOSITIVO SERVER
Identificatore di transazione	2 byte	Identificazione di una transazione di request/response	Inizializzato dal client	Ricopiato dal server dalla request ricevuta.
Identificatore di protocollo	2 byte	0 = protocollo Modbus	Inizializzato del client	Ricopiato dal server dalla request ricevuta.
Lunghezza	2 byte	Numero di byte seguenti	Inizializzato del client (request)	Inizializzato dal server (response)
Identificativo di unità	1 byte	Identificativo di uno slave remoto connesso su una linea seriale od altro bus.	Inizializzato del client	Ricopiato dal server dalla request ricevuta.

Tabella 10 - Campi contenuti nel Modbus Application Protocol header di un messaggio Modbus TCP/IP.

La lunghezza totale dell'header è di 7 byte strutturato secondo i seguenti punti:

- *identificatore di transazione:* è utilizzato per il controllo di parità della transazione, il dispositivo server ricopia questo campo nella response;
- *identificatore di protocollo:* il protocollo Modbus viene identificato dal valore numerico decimale 0;
- *lunghezza:* è il contatore dei byte dei campi successivi, incluso l'identificatore di unità ed i campi dei dati;
- *identificatore di unità:* questo campo è utilizzato per l'instradamento intra sistema del messaggio. Tipicamente utilizzato per comunicare su una linea Modbus Plus o Modbus seriale attraverso un gateway tra una rete Ethernet TCP/IP ed una linea seriale Modbus. Questo campo viene impostato dal client nel messaggio di request e deve essere ricopiato nel messaggio di response del server.

4. Descrizione funzionale

Il modello architetturale si presenta come un modello generale applicabile a qualsiasi dispositivo sia esso un client od un server, alcuni dispositivi però possono operare solamente come client o come server.

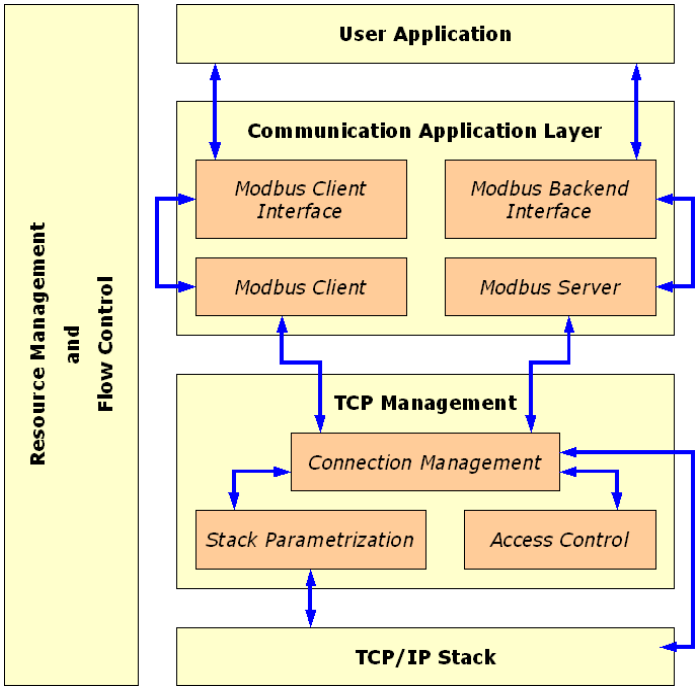


Figura 6 - Modello architetturale del protocollo Modbus TCP/IP.

4.1.Communication Application Layer

Un dispositivo Modbus può fornire un interfaccia Modbus client e/o server ed un'interfaccia backend, permettendo all'utente di accedere indirettamente agli oggetti applicativi. Tale interfaccia è composta da quattro aree: input discreti, output discreti (coils), registri di input, registri di output (tabella 11). I dati dell'applicazione utente andrebbero in ogni caso mappati con tale interfaccia.

TABELLA PRIMARIA	TIPO DI OGGETTO	TIPO	COMMENTO
Input discreti	Singolo bit	Sola lettura	Questo tipo di dato può essere fornito da un sistema di I/O.
Coils	Singolo bit	Lettura – scrittura	Questo tipo di dato può essere alterato da una applicazione.
Registri di input	Parola a 16 bit	Sola lettura	Questo tipo di dato può essere fornito da un sistema di I/O.
Registri di holding	Parola a 16 bit	Lettura – scrittura	Questo tipo di dato può essere alterato da una applicazione.

Tabella 11 - Riassunto delle caratteristiche delle quattro aree che compongono l'interfaccia utente.

Nel livello applicativo possiamo incontrare quattro elementi costitutivi:

- *Client Modbus:* consente all'applicazione utente di controllare lo scambio di dati con un dispositivo remoto. Il client Modbus costruisce una request partendo da dei parametri specificati tramite una *demand* spedita dall'applicazione utente all'interfaccia client Modbus. Il client Modbus utilizza poi una transazione per gestire la tempistica e l'esecuzione dei processi per soddisfare una data request, fornendo poi una confirmation al client stesso.
- *Interfaccia client Modbus:* fornisce un interfaccia che permette all'applicazione utente

di costruire la request voluta utilizzando servizi Modbus differenti, incluso l'accesso agli oggetti applicativi.

- *Server Modbus:* alla ricezione di una request Modbus questo modulo attiva una azione locale per leggere, scrivere od eseguire qualche altra azione. L'esecuzione di queste azioni è effettuata in modo totalmente trasparente per il programmatore. Le funzioni principali del server Modbus sono di attendere una request Modbus sulla porta TCP 502, soddisfare questa request e poi costruire una response pertinente alla richiesta effettuata.
- *Interfaccia backend Modbus:* è un'interfaccia dal server all'applicazione server in cui vengono definiti gli oggetti applicativi.

4.2.TCP management layer

Una delle principali funzioni del servizio di messaggistica fornite da Modbus TCP/IP è quella di gestire la comunicazione, in modo di stabilire un termine della comunicazione e gestire il flusso dei dati sulle connessioni TCP. Anche in questo caso possiamo distinguere tre componenti principali:

- *Connection management:* il processo di comunicazione tra un client ed un server Modbus richiede l'uso di un modulo di gestione della connessione TCP: tale modulo permette di gestire in modo globale il meccanismo di messaggistica delle connessioni TCP, secondo due differenti modalità. Per le comunicazioni attraverso il protocollo Modbus TCP/IP viene dedicata una porta particolare ossia la numero 502, che viene monitorata di default dal sistema. Alcune applicazioni possono richiedere che venga dedicata un'altra porta alla comunicazione Modbus TCP/IP, per questa ragione si raccomanda che sia il client che il server abbiano la possibilità di permettere all'utente di impostare un numero di porta personalizzato. È importante ricordare che sebbene vi sia un porta aggiuntiva dedicata alla comunicazione, la numero 502 rimane in ogni caso dedicata al protocollo Modbus TCP/IP.
  - *Gestione diretta delle connessioni:* in questo caso è l'applicazione utente che gestisce direttamente la connessione TCP. Questo tipo di connessione viene effettuato sia per il dispositivo client che per il dispositivo server utilizzando un socket BSD per gestire la connessione. Questo tipo di implementazione offre maggiore flessibilità ma implica una maggiore "responsabilità" da parte del programmatore poiché richiede una certa conoscenza delle specifiche TCP.
  - *Gestione automatica delle connessioni:* la gestione della connessione TCP è totalmente trasparente all'applicazione utente. Il modulo di gestione delle connessioni gestisce un numero sufficiente di connessioni client – server, in ogni caso deve essere implementato un un meccanismo di gestione del sovrannumero di connessioni il quale chiude le connessioni più vecchie ed inutilizzate. Una connessione viene aperta se e solo se un dispositivo riceve un pacchetto da un client o da un'applicazione. Tale connessione viene chiusa se si invia un pacchetto di chiusura dalla rete dal dispositivo stesso. Alla richiesta di connessione il modulo di controllo di accesso permette di filtrare gli accessi dei client non autorizzati. Per mantenere la compatibilità tra sistemi e risorse server si utilizzano due pool di connessioni:

- *Priority connection pool:* formato dalle connessioni che non sono mai chiuse per iniziativa locale. Per implementare tale pool si associano gli indirizzi IP ad ogni connessione del pool. Il dispositivo avente un indirizzo IP incluso nel pool è chiamato "marked" ed ogni connessione richiesta dal dispositivo marcato deve essere accettata. Per migliorare l'efficienza del pool delle connessioni deve essere impostato un parametro che consente di gestire il numero massimo di connessioni.
- *Non – priority connection pool:* contiene le connessioni per i dispositivi non marcati. La regola di gestione di tali connessioni consiste nel chiudere le connessioni più vecchie quando un dispositivo non marcato richiede una connessione e non vi sono più connessioni disponibili nel pool.

- *Modulo di controllo degli accessi:* in certi contesti critici l'accessibilità a dati interni dei dispositivi deve essere revocata per determinati host, con questo modulo è possibile ovviare a tale incombenza, consentendo in tal modo di controllare gli accessi a determinati oggetti e realizzare applicazioni sicure. Mediante una lista di indirizzi IP il modulo controlla ogni nuova connessione autorizzando o meno l'accesso ai dispositivi. Nel caso venga richiesto l'accesso da parte di un dispositivo anonimo senza indirizzo IP, la connessione viene automaticamente chiusa.
- *Parametrizzazione dello stack:* mediante opportuni parametri del layer TCP/IP è possibile modificare il comportamento globale del sistema. Esistono parametri per la gestione del flusso dei dati, della dimensione del buffer di ricezione dei dati, del tempo di delay, di keep – alive e di impostazione degli indirizzi di sottorete e di default gateway.

4.3.TCP/IP stack layer

Lo stack TCP/IP può essere parametrizzato in modo da poter modificare il controllo del flusso di dati, la gestione degli indirizzi e la gestione delle connessioni definendo differenti vincoli specifici per ogni dispositivo o sistema. Per equilibrare il flusso di dati tra il client ed il server viene fornito un meccanismo di controllo di flusso dei dati su tutti i livelli dello stack di messaggistica Modbus attraverso un modulo di gestione delle risorse e del flusso di sistema. Tale modulo è basato sul flusso di controllo interno di TCP/IP in aggiunta al controllo al livello data link ed al livello applicativo utente.

CAPITOLO 3 - IL PROTOCOLLO MODBUS PLUS

1. Introduzione

Il protocollo Modbus Plus permette di realizzare reti locali che attuano il controllo distribuito dei dispositivi industriali di I/O. In questa versione del protocollo si fa uso di una serie di dispositivi di proprietà della Modicon, chiamati *Modicon TSX Quantum Automation controller* i quali sono compatibili con un ampia gamma di dispositivi industriali.

Una rete implementata attraverso il protocollo Modbus Plus fa uso di cinque tipologie di dispositivi:

- Un dispositivo appartenente alla serie *Modicon TSX Quantum Automation controller* su cui è presente il programma applicativo ed è realizzata la porta fisica Modbus Plus, la quale consente al dispositivo di operare come master della rete di comunicazione.
- Fino a due dispositivi chiamati *Modbus Plus Network Module (NOM)* installati sulla backplane[1] della rete, i quali operano come dispositivi master per le rispettive sottoreti.
- Dei dispositivi chiamati *Distribuited I/O Drop adapters (DIO)* anch'essi installati sulla backplane della rete che permettono ognuno il controllo di 15 dispositivi di I/O, e forniscono la potenza operativa dei dispositivi stessi attraverso il collegamento con la backplane.
- Dei dispositivi chiamati *Terminal Block I/O (TIO)* installati sui pannelli dei dispositivi remoti, che possono essere connessi direttamente ai dispositivi di I/O.
- Dei moduli *Modicon ModConnect,* prodotti *third party* (ossia aziende esterne alla Modicon) ed installati come dispositivi remoti.

Nella seguente figura 7 è possibile osservare la schematizzazione di una rete realizzata mediante la tecnologia Modbus Plus e composta da tre reti. Una di queste è connessa alla porta Modbus Plus della CPU le restanti due reti sono connesse alla backplane di quest'ultima attraverso i rispettivi dispositivi NOM.

1 *Backplane:* con tale termine si indicano tutti quei circuiti stampati che consentono di connettere dei connettori in parallelo ad altri connettori, in modo tale che il pin di un connettore sia collegato al medesimo pin di un altro connettore. Una backplane permette inoltre di realizzare dei sistemi di elaborazione composti da più circuiti stampati.

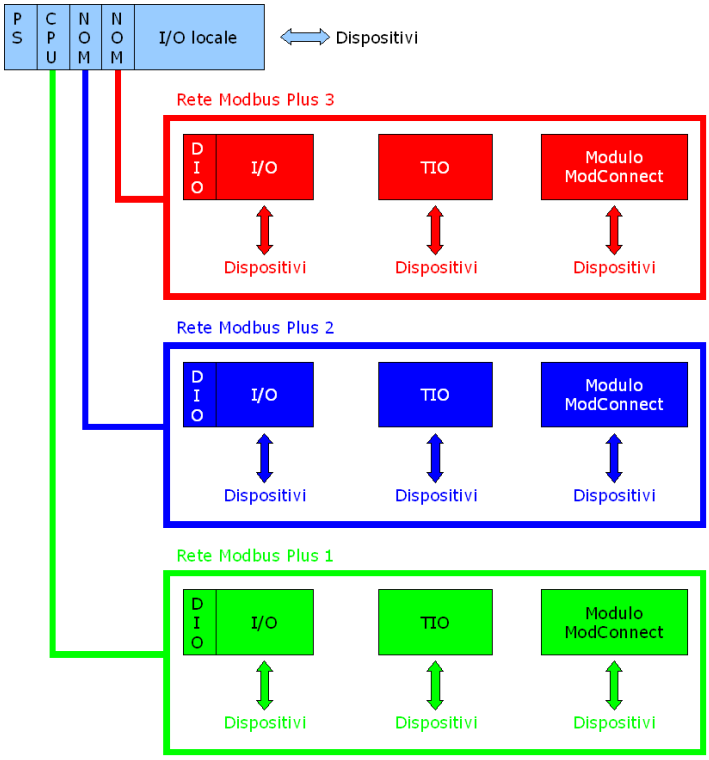


Figura 7 - Schematizzazione di una rete realizzata mediante il protocollo Modbus Plus.

2. La rete

Abbiamo visto nella precedente figura 7 e nell'introduzione che una rete Modbus Plus può essere formata da un massimo di tre sottoreti ognuna costituita fino ad un massimo di 64 dispositivi detti *nodi*. La rete può essere connessa direttamente alla porta Modbus Plus di un dispositivo indirizzato attraverso un unico indirizzo di rete, in tal caso il dispositivo chiamato *controller di rete* supporta fino ad un massimo di 63 nodi di I/O.

Nell'introduzione abbiamo visto inoltre la possibilità di utilizzare fino a due NOM ognuno indirizzabile attraverso il proprio indirizzo di rete e supportante anch'esso fino a 63 nodi di I/O: una rete che utilizzi un controller e due NOM può quindi connettere fino a 189 nodi.

I dispositivi controller Modicon ed i NOM sono configurabili attraverso otto cavi singoli o duali a seconda del layout di rete che si vuole ottenere. Utilizzando un cavo della lunghezza di 450 m è possibile connettere fino a 32 nodi, ed attraverso dei ripetitori e dei cavi della lunghezza di 1800 m, tale rete può essere estesa fino a 64 nodi, inoltre il supporto della fibra ottica consente la realizzazione di reti aventi distanze maggiori.

3. I dispositivi di rete

Rispetto al protocollo Modbus seriale e TCP/IP, le reti Modbus Plus utilizzano una particolare terminologia che costituisce l'eterogeneità dei dispositivi con cui si possono implementare le reti.

- *Nodo:* con questo termine si identifica un qualsiasi dispositivo connesso ad una rete Modbus Plus. Alcuni dispositivi come i controller programmabili, i NOM, gli adattatori drop DIO ed i moduli TIO hanno un indirizzo di rete che viene utilizzato per identificare il dispositivo come sorgente o destinazione di un messaggio. Esiste

inoltre un nodo senza indirizzo chiamato *ripetitore* utilizzato per estendere una rete Modbus Plus.

- *Nodo I/O master:* sempre presente su una rete Modbus Plus, può essere un controller programmabile od un NOM. I nodi master possono spedire o ricevere dati dai nodi drop.
- *Nodo I/O drop:* con questo termine si intendono gli adattatori DIO drop ed i moduli TIO. I nodi I/O drop possono ricevere dati da un nodo master e possono spedire dati solo allo stesso nodo master.
- *Segmento di cavo:* è una unità di un cavo trunk compresa tra due prese. Le prese sono dispositivi passivi che forniscono le connessioni per i segmenti di cavo e per i cavi di drop destinati ai nodi I/O drop. Su una rete con layout a cavo duale due segmenti di cavo corrono in parallelo tra due prese a due distinti I/O drop.
- *Sezione:* corrisponde ad una serie di nodi connessi tramite segmenti di cavo, con associate le rispettive prese e cavi di drop. Il percorso del segnale di sezione non deve passare attraverso nessun dispositivo e le sezioni sono tutte parti di una stessa rete che condivide lo stesso token e la stessa sequenza di indirizzi. Ogni sezione può raggiungere una lunghezza massima di 450 m e collegare fino a 32 nodi. Due sezioni possono essere connesse attraverso un ripetitore per estendere sia la lunghezza del cavo che il numero di nodi.
- *Rete:* si intende l'insieme dei nodi posti sullo stesso percorso di un segnale, accessibile mediante la rotazione del token. Può essere formata da una o più sezioni, nella cui configurazione standard è composta da un controller (nodo master) ed uno o più nodi I/O drop. Oltre al controller di rete possono essere installati fino a due NOM permettendo all'applicazione utente di interagire con un totale di tre reti.
- *Drop:* forniscono collegamenti per i cavi a dispositivi nodo ed al terminale di terra.
- *Presa:* forniscono il collegamento dei nodi alla rete mediante connessioni "a tunnel", permettendo al cavo trunk di passare attraverso la presa. Inoltre fornisce una connessione drop per i dispositivi ed una connessione al terminale di terra.
- *Cavo trunk*[2]*:* è un cavo incrociato e schermato che copre il percorso tra due nodi successivi. La lunghezza di tale cavo va dai 3 ai 450 metri, nelle reti che sfruttano il layout con cavo duale i cavi vengono identificati come cavo A e B. Ogni cavo duale può raggiungere la lunghezza di 450 m misurata tra due nodi di una stessa sezione e la differenza di lunghezza tra il cavo A e B non deve superare i 150 m nella stessa sezione.

4. I layout di rete

Le reti Modbus Plus possono avere due tipologie di layout a seconda del tipo di cavo impiegato e della quantità di nodi connessi.

Nella figura 8 è possibile osservare una rete con layout di base composta da *n dispositivi* fino ad un massimo di 32 nodi, mentre nella figura 10 è possibile osservare la medesima

2 *Cavo trunk:* è un cavo a fibra ottica che fornisce il collegamento per pipeline, circuitale ed a canale per una connessione trunk, in una rete che sfrutta la comunicazione a fibra ottica. Consente di avere da 4 a 96 fibre ottiche per trunk e distribuire dati su canali multipli.

rete estesa a 64 nodi mediante un ripetitore. In questa seconda topologia di rete il ripetitore può essere sostituito da una coppia di ripetitori collegati tra di loro mediante fibra ottica: ciò permette di implementare reti con un'estensione di addirittura 2 – 3 km, dipendente dalla sezione del cavo impiegato.

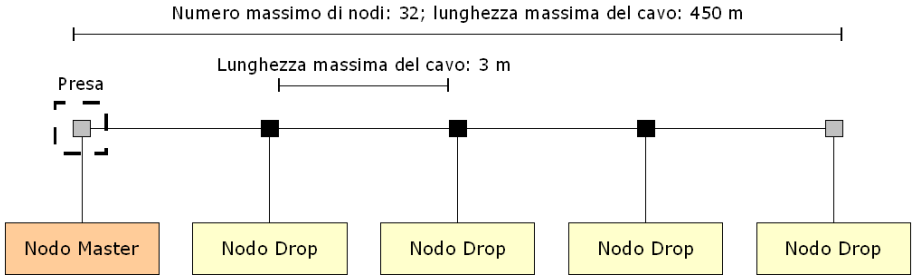


Figura 8 - Schematizzazione di una rete Modbus Plus che sfrutta il layout di rete di base.

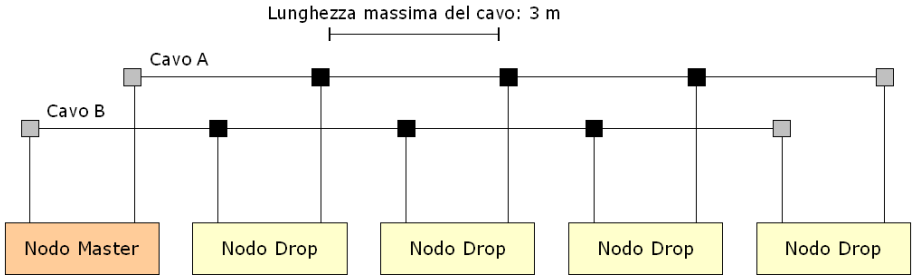


Figura 9 - Schematizzazione di una rete Modbus Plus che sfrutta il layout di rete con cavo duale.

La figura 9 illustra invece un layout di rete che sfrutta il collegamento duale del cavo, questa topologia di rete offre un certo grado di protezione contro i guasti dovuti alle connessioni di uno dei due cavi e permette in caso di emergenza, l'instradamento del segnale sul cavo alternativo.

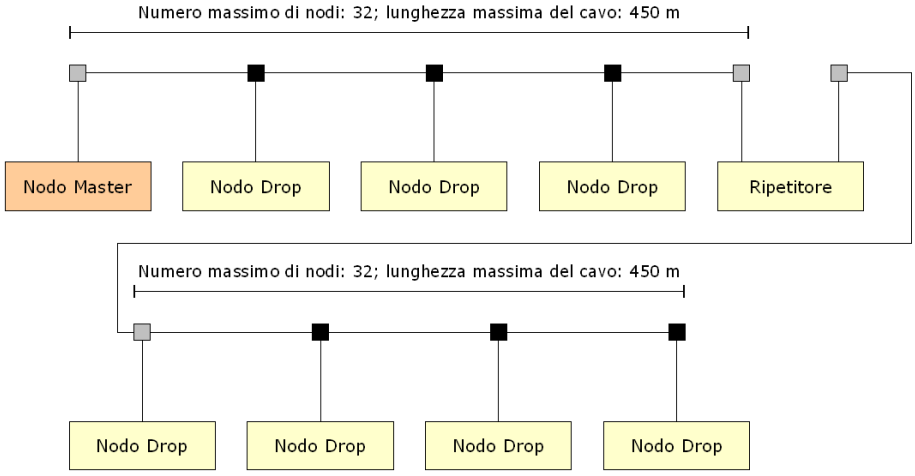


Figura 10 - Schematizzazione di una rete Modbus Plus che sfrutta il layout di rete con ripetitore.

5. Funzionamento di una rete Modbus Plus

I nodi della rete sono identificati da indirizzi assegnati dall'utente, tali indirizzi sono assegnati impostando degli interruttori siti in ogni nodo. L'indirizzo del nodo è indipendente dalla sua ubicazione nella rete ed è rappresentato da un valore numerico decimale che va dal valore 1 al valore 64 senza che sia necessaria la sequenzialità dei valori all'interno della

rete: l'unico vincolo a tali valori è che non vengano ripetuti all'interno della stessa rete.

I nodi della rete funzionano come *membri peer* di un anello logico, guadagnando il diritto di eseguire un azione mediante la ricezione di un *token*. Il token è rappresentato da un gruppo di bit passati in sequenza rotativa da un nodo ad un altro. In applicazioni che utilizzano più reti, ogni rete mantiene la propria sequenza di rotazione indipendentemente dalle altre ed il token non può essere scambiato tra le reti.

Oltre al proprio indirizzo un nodo deve essere riconosciuto dall'applicazione utente. Mediante la configurazione iniziale un controller attraverso il pannello software, configura la *DIO Map* e la tabella *Peer Cop* mediante cui vengono identificati gli indirizzi dei vari nodi della rete ed i rispettivi riferimenti logici per il trasferimento dei dati.

6. La sequenza di rotazione del token

La sequenza di rotazione dei token è determinata dall'indirizzo dei nodi: la rotazione inizia dal nodo con indirizzo più basso passando in modo crescente all'indirizzo più alto. Al termine della rotazione, ossia al nodo con indirizzo più alto, il token viene riassegnato al nodo con l'indirizzo più basso della rete. Se un nodo lascia la rete, in 100 millisecondi la sequenza del token viene aggiornata, se invece si aggiunge un nodo alla rete, l'aggiornamento della sequenza avviene entro cinque secondi. Il processo di rimozione ed aggiunta dei nodi è trasparente all'applicazione utente ma influisce sull'operatività dei dispositivi di I/O.

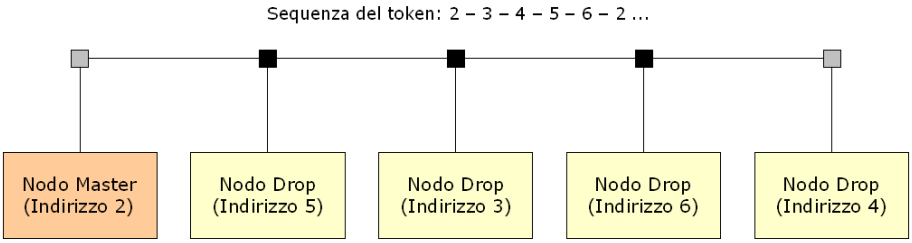


Figura 11 - Esempio di sequenza di rotazione dei token all'interno di una rete.

7. Rilevamento dei nodi da parte dell'applicazione utente

Nel precedente paragrafo 5 abbiamo visto che prima di eseguire l'applicazione utente, il controller di rete esegue il processo di configurazione della stessa, mediante una serie di strumenti hardware e software. Tramite speciali pannelli chiamati Modicon Modsoft e ConCept nel processo di configurazione alla *Dio Map* vengono identificati i controller, i nodi drop, ed i NOM dotati di adattatori DIO Drop. Mediante la tabella *Peer Cop* vengono invece identificati i controller, i nodi drop, ed i NOM dotati di moduli TIO.

La tecnica di mappatura DIO è la medesima della mappatura dei dispositivi remoti di I/O (*Remote I/O*, *RIO*) in cui vengono specificati sia il dispositivo terminale di testa (master) che il dispositivo terminale di drop (il dispositivo remoto).

Durante il processo di configurazione il controller ed i dispositivi NOM vengono identificati mediante la loro posizione della backplane ed al controller viene assegnato l'identificativo numerico 0. Gli adattatori DIO ed i moduli TIO vengono identificati mediante un identificativo corrispondente alla combinazione della posizione del loro controller o NOM e dell'indirizzo di rete.

8. Configurazione dei dispositivi DIO

All'avvio della rete il controller scarica la DIO Map in ogni adattatore DIO drop in modo diretto od attraverso il relativo NOM locale. La DIO Map dell'adattatore Drop è una sezione della DIO Map del controller che riguarda il singolo drop. Ogni adattatore Drop memorizza la propria DIO Map e mediante questo processo vengono impostati i privilegi all'interno dell'adattatore drop in modo da permettere che solo il nodo master possa scaricare la DIO Map in ogni nodo. I successivi tentativi da parte di altri nodi di sovrascrivere la DIO Map dell'adattatore drop, falliscono mentre se il nodo master viene rimosso dalla rete l'adattatore drop accetta una nuova DIO Map dal nuovo master assegnato. Quando la DIO Map dell'adattatore drop viene ricevuta, il drop comincia a spedire i messaggi ricevuti al prossimo nodo che ha ricevuto il token. Se un drop non viene configurato in modo tale da poter spedire dati, in ogni messaggio spedisce due parole per il controllo della checksum e dello status della DIO Map.

9. Configurazione dei moduli TIO

Attivando la rete il controller inizia la gestione dei moduli TIO attraverso la tabella Peer Cop. Le transazioni vengono gestite secondo la sequenza degli indirizzi di rete, con il controller od i NOM, a cui viene assegnata la posizione del nodo con indirizzo più basso, passando poi alla gestione della transazione attraverso i nodi TIO e trasmettendo i dati nell'ordine con cui si riceve il token.

Ogni modulo di output è dotato di un registro che contiene un parametro di temporizzazione per mantenere il valore di stato corrente dell'output, nel caso in cui il modulo non sia aggiornato da un nuovo comando di scrittura. Se il tempo di mantenimento scade, l'output del modulo si attiva al valore logico 0. Tale intervallo di tempo può variare da un minimo di 300 ms ad un massimo di 60 s con incrementi di 10 ms: il tempo di default impostato è tipicamente di un secondo.

10.I privilegi di scrittura degli adattatori DIO drop

Dopo il caricamento della DIO map nell'adattatore drop, e la ricezione da parte di quest'ultimo di un messaggio dati, l'adattatore drop può essere oggetto di due tipi di privilegi di scrittura: il privilegio di scrivere una nuova DIO Map nel drop, ed il privilegio di scrivere nuovi dati nel drop.

Al nodo master sono concessi entrambi i privilegi: nel caso in cui tale nodo scrive la DIO Map nell'adattatore, esso diverrà l'unico dispositivo in grado di scrivere anche i dati. Gli eventuali altri nodi master della rete possono avere solamente i diritti di lettura dei messaggi contenuti nell'adattatore drop. Per quanto riguarda il numero di nodi master che possono leggere un particolare drop non sussiste nessuna restrizione. Una singola rete I/O può infine contenere un numero imprecisato di nodi mater che leggono i dati dai dispositivi drop, ma sempre e solo un nodo master ha il diritto di scrivere i dati.

11.I privilegi di scrittura dei moduli TIO

Quando un nodo master scrive su un modulo TIO tale diritto rimane attivo per 60 secondi, e tale diritto può essere mantenuto reiterando ogni 60 secondi la scrittura. Allo scadere dei 60 secondi il modulo può accettare la scrittura da parte di un altro nodo.

L'intervallo di tempo di 60 secondi è fisso e non modificabile dall'applicazione. Tale processo di timeout è applicabile solamente alla scrittura mentre la lettura degli input o dei dati del modulo è accessibile da parte di qualsiasi nodo della rete.

12.La gestione dei messaggi di rete

La sorgente dei dati e la destinazione per gli adattatori DIO drop sono specificati dalla DIO Map scaricate nel drop all'avvio della rete. La sorgente dei dati e di destinazione per i moduli TIO viene invece specificata dalla tabella Peer Cop del controller.

Quando il controller od il NOM di rete riceve il token, viene trasmesso all'adattatore drop per recapitarlo al modulo di output, ed al modulo TIO per recapitarlo ai punti di output. Quando il token viene recapitato all'adattatore drop, i dati vengono trasmessi dai moduli di input al controller o NOM di rete. Quando il token viene invece recapitato al modulo TIO i dati vengono recapitati dal punto di input al controller o NOM.

I messaggi instradabili su una rete Modbus Plus sono quantità definite di parole a 16 bit; il numero di parole oggetto di una transazione varia dal tipo di dispositivo coinvolto (nodo master, adattatori DIO drop o moduli TIO). I nodi master possono trasmettere e ricevere fino a 500 parole di dati, numero massimo di parole che si possono specificare nella configurazione di un controller. I nodi adattatore DIO drop possono ricevere fino a 32 parole di cui ognuno può contenere dati per i moduli dei nodi di output. I nodi adattatore drop possono trasmettere fino a 32 parole per messaggio. Le prime due parole sono dedicate alle variabili di stato del nodo DIO drop mentre le restanti parole contengono i dati provenienti dai moduli di input. I moduli TIO possono infine ricevere e trasmettere 2 o 3 parole nei messaggi come specificato dal tipo di modulo.

CAPITOLO 4 - IL PROTOCOLLO MODBUS ENRON

1. Introduzione

Oltre alle versioni fornite da Modicon alcune aziende di produzione di dispositivi hardware immettono sul mercato proprie versioni di Modbus: una di queste ad esempio è il protocollo Modbus Enron, una delle tante varianti del protocollo Modbus seriale.

Modbus Enron è stato sviluppato dalla Enron Corporation e la differenza principale tra questo protocollo ed il suo parente prossimo, il protocollo Modbus seriale, consiste nella numerazione degli indirizzi di registro, il supporto di registri a 32 bit come registri a 16 bit, e la possibilità di spedire eventi di log e dati di tipo storico.

2. Tipi di dato in Modbus Enron

Per ogni tipo di dato esiste una tabella dedicata (tabella 12) e l'informazione è immagazzinata nel dispositivo slave in quattro differenti tabelle: una di queste memorizza valori discreti (booleani) mentre le altre memorizzano valori numerici. Ogni valore nella tabella ha un indirizzo di dato ed un rispettivo numero di registro ed a differenza di Modbus seriale, nel protocollo Modbus Enron il numero di registro è uguale all'indirizzo dati.

INDIRIZZO DATI (ESADECIMALE)	NUMERO DI REGISTRO (DECIMALE)	TIPO	NOME DELLA TABELLA
03E9 – 07CF	1001 – 1999	Lettura – scrittura	Variabili booleane
0BB9 – 0F9F	3001 - 3999	Lettura – scrittura	Variabili short integer a 16 bit
1389 – 176F	5001 - 5999	Lettura – scrittura	Variabili long integer a 32 bit
1B59 – 1F3F	7001 - 7999	Lettura – scrittura	Variabili floating point a 32 bit

Tabella 12 - Tipi di dato disponibili nel protocollo Modbus Enron.

Come nel protocollo standard anche in Modbus Enron il master utilizza dei codici per comandare al dispositivo slave di eseguire una determinata azione su un certo insieme di dati (tabella 13).

CODICE FUNZIONE	AZIONE	NOME E NUMERO DELLA TABELLA
01	Lettura	Variabili booleane (1000)
05	Set single	Variabili booleane (1000)
03	Lettura	Variabili numeriche (3000, 5000, 7000)
06	Set single	Variabili numeriche (3000, 5000, 7000)

Tabella 13 - Esempi di codici funzione utilizzati dal dispositivo master in una comunicazione che sfrutta il protocollo Modbus Enron.

3. Gli eventi

Nel protocollo esistono due tipi di evento: gli eventi operatore e gli eventi allarme. Gli *eventi operatore* sono modifiche ad opportuni oggetti preventivamente mappati, nel caso in cui si voglia registrare un evento alla modifica del valore associato, anche il valore va incluso nella mappatura. Esiste però un lato negativo di questa caratteristica del protocollo ossia che non è possibile definire quale oggetto mappato abbia creato un particolare evento operatore poiché tutti gli oggetti mappati, alla loro modifica, creano eventi. Con *eventi allarme* si intendono invece tutti quegli eventi che si verificano quando il valore di un oggetto mappato eccede i limiti prestabiliti definiti nel dispositivo slave.

Il dispositivo slave è in grado di memorizzare almeno i cento eventi più recenti indistintamente tra eventi operatore ed allarme. Un valore numerico chiamato *indice di evento* è tipicamente mappato tramite la variabile numerica 7000 in modo tale da poter essere letto e corrisponde al numero di eventi che si sono verificati dall'ultima volta che il buffer degli eventi è stato letto.

Nel caso il dispositivo master interroghi il dispositivo slave, quest'ultimo conterrà tutti gli eventi che si sono verificati da quando sono stati collezionati l'ultima volta. Se si sono verificati più di 255 eventi, sono i primi 255 vengono trasmessi al dispositivo master. Nel caso in cui non si siano verificati eventi, lo slave manderà una risposta senza nessun valore nel campo dati.

Dopo la trasmissione di un evento il master può spedire un *messaggio di acknowledgment* allo slave in modo tale da resettare l'indice degli eventi e pulire il log ad essi associato.

4. Trasmissione dei dati storici

Nel protocollo Modbus Enron esistono due tipi di tabelle storiche: la *tabella giornaliera* e la *tabella oraria*, le cui dimensioni sono impostate nel dispositivo slave. Gli archivi orari e giornalieri sono liste di valori di oggetti storici dotati di timestamp e tali oggetti sono definiti nella mappatura del protocollo Modbus.

Nell'archivio giornaliero vengono registrati i valori per gli oggetti storici alla fine di ogni giorno, mentre nell'archivio orario si registrano i valori alla fine di ogni ora. Prendendo come riferimento un periodo di tempo uguale, un archivio orario contiene ovviamente un numero di registrazioni ventiquattro volte superiore rispetto ad un archivio giornaliero.

Ad ogni record è fornito un numero identificativo, tale numero viene impostato a zero quando la tabella di archivio è piena ed il dispositivo deve quindi sovrascrivere i vecchi dati. I valori più recenti vengono memorizzati come variabili chiamate *indice orario*, per l'archivio

orario ed *indice giornaliero* per l'archivio giornaliero. Questi variabili indice sono tipicamente incluse nella mappatura come variabili numeriche della serie 7000 in modo tale da poter essere successivamente letti.

BIBLIOGRAFIA

Modicon, **Modicon Modbus Plus Network I/O Servicing Guide**, http://www.alamedaelectric.com/Modicon%20Documents/PLC%20MB+%20Networking%20IO%20Servicing%20Guide%20v2.0.pdf, 1996

Modbus-IDA, **Modicon Modbus Protocol Reference Guide**, http://www.modbus-ida.org, 1996

Modbus-IDA, **Modbus messagging on TCP/IP implementation guide V1.0b**, http://www.modbus-ida.org, 2006

Modbus-IDA, **Modbus application protocol specification V1.1b**, http://www.modbus-ida.org, 2006

Simply Modbus, **Enron Modbus**, http://www.simplymodbus.ca, 2008

Wikipedia, **Wikipedia**, http://www.wikipedia.org, 2008