AOX

SOFTWARE DEL KIT IEC61131-3

18 June 2011

Premessa

Con questa terza puntata della serie **Un KIT programmabile IEC61131-3** concluderò la realizzazione del PLC didattico. Nelle prime due puntate mi sono occupato della **progettazione hardware** e della **costruzione del prototipo**. Ora è giunto il momento di dare corrente alla scheda e di pensare alla **creazione del software**. Se fosse stata una scheda dedicata ora avrei dovuto prendere tutte le specifiche funzionali della macchina o dell'impianto d'automazione e poi realizzare l'intero programma applicativo, magari utilizzando il linguaggio C. Ma il mio obiettivo è di realizzare un software, unico ed indipendente dall'applicazione, che trasformi la mia scheda in un qualcosa di programmabile da altri, utilizzando il linguaggio **standard IEC**. Per ora la scheda è un qualsiasi hardware a microprocessore, ma per poterla utilizzare come un PLC è necessario creare un programma che, in accordo con l'ambiente di sviluppo su PC, in questo caso **CoDeSys**, esegua un qualsiasi programma applicativo scritto in linguaggio IEC.
Voglio far presente ai lettori che questo blog è un **mattone** essenziale per terminare la costruzione del PLC didattico, prima di passare a semplici e più piacevoli esercizi di programmazione mediante CoDeSys. Prometto che il prossimo blog sarà ben più leggero perchè mi dedicherò solo a **giocare col PLC**.

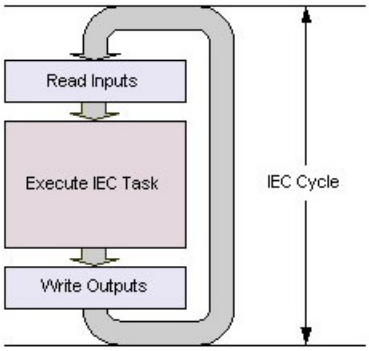
Cosa manca per utilizzare la scheda come PLC

Voglio ricordare la distinzione che esiste tra una scheda a microprocessore ed un PLC, argomento già da me trattato nel blog **Elettronica nell'automazione industriale**. Spesso si utilizza il termine PLC per indicare una scheda a microprocessore che, in quanto dotata di ingressi ed uscite, viene utilizzata per automatizzare una macchina. Questa scheda non può operare come un PLC perché non ha un sistema operativo, un ambiente di sviluppo ed un linguaggio di programmazione per poterla considerare tale. Il PLC era nato per consentire a personale, anche non esperto di elettronica, di automatizzare una macchina, sostituendo complessi **impianti a relè cablati**. Dunque il PLC deve essere dotato di tutto il supporto necessario ad utilizzarlo in questo modo. Ciò richiede un software preinstallato sulla scheda che traduca in concreto le funzioni d'automazione programmate. Per questo è anche necessario uno strumento di sviluppo e di programmazione dell'applicativo, che ormai coincide quasi sempre con un software

su Personal Computer.
Quindi per utilizzare il mio hardware come un PLC servono gli ultimi due elementi: il **software di funzionamento come PLC** ed il **software di sviluppo su PC**. Il software da installare sulla scheda verrà chiamato nel seguito **RTS** (Run Time System), mentre il software installato sul PC sarà **CoDeSys**. Questo ambiente di sviluppo è indipendente dalla marca e dal tipo di apparecchio, tanto che può essere utilizzato su PLC diversi, schede a microprocessore ed anche PC industriali. Tuttavia CoDeSys deve sapere con chi ha a che fare e per questo è necessario installare sul PC anche un KIT di files per ogni particolare apparecchio usato. Questo insieme di files, detto **TSP** (Target Support Package) descrive a CoDeSys le caratteristiche dello specifico PLC, chiamato genericamente **Target**. Questa descrizione comprende la configurazione particolare degli ingressi ed uscite, la loro mappatura in memoria, le librerie IEC delle specifiche risorse hardware, i files di help on-line da integrare a quello standard ed altre informazioni.

L'esecuzione di un programma IEC sul PLC

Per molti anni la grande diffusione dei PLC, prodotti da più case costruttrici, è stata caratterizzata da una grande varietà dei relativi linguaggi di programmazione, spesso simili, ma non abbastanza da poterli considerare compatibili tra loro. Successivamente la **International Electro-technical Commission** (in breve IEC) ha gettato le basi per una standardizzazione del linguaggio dei PLC. Ovviamente un unico linguaggio standard ha notevoli vantaggi per l'utilizzatore, mentre per alcuni grandi produttori di PLC questo può costituire un limite al vincolo col cliente. Tuttavia il **linguaggio IEC61131-3** non è stato inventato da zero e controcorrente, ma è il risultato delle passate esperienze anche dei grandi produttori.
La definizione di un linguaggio d'automazione standard e la necessità di creare software riciclabile ha portato alla scomposizione del programma in unità più elementari dette **POU** (Program Organization Unit). Una POU può essere un modulo di programma, un blocco funzionale o una semplice funzione. L'intero applicativo è costituito da almeno una POU di tipo programma, che viene scansionata dalla CPU in sequenza con l'aggiornamento degli ingressi e delle uscite utilizzati nella POU:



Rappresentazione di un ciclo IEC

Gli ingressi sono fisicamente letti dai morsetti della scheda e copiati nella relativa area di memoria immagine. Poi i valori degli ingressi e di altre memorie di lavoro sono combinati tra loro, secondo l'esecuzione delle istruzioni del programma applicativo, definendo così il valore aggiornato dell'area di memoria immagine delle uscite. Infine il valore delle uscite è fisicamente copiato dalla memoria sulle interfacce di uscita (ad esempio i relè). Tutto questo ciclo completo può avvenire a scadenze regolari, oppure continuamente, oppure sotto certe condizioni od eventi esterni.
Nel primo caso di scadenze regolari, la POU del programma IEC è associata ad una **Task periodica** il cui tempo di ripetizione (ad esempio ogni 10ms) e la priorità relativa alle altre Task è stabilita dal programmatore.
Nel secondo caso il ciclo completo Ingressi-Elaborazione-Uscite è eseguito in una **Task freewheeling** (a ruota libera) cioè ripetutamente, uno dopo l'altro, senza aspettare nessuna scadenza.
Nel terzo caso il ciclo è inserito in una **Task evento** ed eseguito di conseguenza al verificarsi di una particolare situazione, come la variazione di un segnale proveniente dagli ingressi digitali esterni.
In generale l'intero applicativo è composto da più POU di tipo programma raggruppate in una o più Task, ciascuna con le sue caratteristiche di attivazione della scansione. Purtroppo per il sistema operativo del PLC si tratta di aggiornare tutti gli ingressi ed uscite di un ciclo IEC in sincronia con la scansione del listato programma dello stesso (o meglio della cascata di tutte le POU inserite nella Task). Questo comporta una rilettura degli ingressi ed una riscrittura delle uscite in un certo senso sovrabbondante ma necessaria. Per esempio se abbiamo una parte di programma scansionata in una Task a 15ms ed un'altra parte, meno critica in tempistica (per esempio relativa alla gestione di un display o di un HMI) scansionata in un'altra Task a 100ms, non è pensabile unificare la lettura degli ingressi per due eventi a periodicità così diversa. Ogni ciclo IEC deve avere tutti i suoi ingressi aggiornati prima della scansione del listato e deve aggiornare tutte le relative uscite dopo la sua scansione. Inoltre ci possono essere degli scambi di valori e dati tra POU programma associate a Task differenti e questo richiede un'accurata gestione di semafori software per evitare di scambiarsi dati non integri. Per quanto appena detto è bene quindi fare un'accurata scomposizione del programma IEC in moduli o POU da inserire nel minor numero possibile di Task differenti.

La creazione del software RTS

La creazione del software RTS (la parte installata sul PLC affinchè operi come tale) avviene mediante la compilazione di un **progetto in linguaggio C**, utilizzando un apposito ambiente di sviluppo. Il progetto è supportato da una libreria di funzioni standard corrispondente all'operazione di porting (adattamento) specifico del processore verso il tool CoDeSys. Questa libreria standard viene poi combinata, dal

linker dell'ambiente di sviluppo, con la parte creata in C per il particolare hardware.
Per esempio, in riferimento al precedente paragrafo, uno dei principali compiti nella creazione del software RTS è quello di sviluppare le funzioni di **lettura degli ingressi** e di **scrittura delle uscite**. Per il progetto in questione, occorre implementare una funzione di aggiornamento degli ingressi che viene chiamata automaticamente ogni volta che inizia un ciclo IEC abbinato ad una Task. Nel nostro caso abbiamo **8 ingressi digitali** ottenuti mediante un IO expander (nello schema IC7) connesso alla porta I2C. La funzione da implementare ci fornisce come parametro il puntatore dell'area di memoria immagine degli ingressi sulla quale dobbiamo posizionare il valore letto da IC7. Per la gestione del I2C si utilizzano delle funzioni API disponibili nella libreria del processore. Quindi la funzione si riduce a poche righe di codice C. In modo analogo occorre implementare la funzione di aggiornamento delle **8 uscite digitali** prendendo, mediante un puntatore passato come paramero della funzione, il valore corrente del byte ed inviandolo all'integrato IC5 tramite sempre la porta I2C.
Con una scheda dotata di un maggior numero di IO avremmo potuto utilizzare altri integrati IO expander con **porta I2C** ed aggiungere le relative istruzioni nelle funzioni di aggiornamento degli ingressi e delle uscite. La scheda è nata con 8 ingressi ed 8 uscite a bordo e quindi la loro gestione è stata incorporata definitivamente nel programma RTS, rendendo tali IO sempre pronti e disponibili alla programmazione IEC. Tuttavia la porta I2C è riportata anche su un connettore per cavo flat a 10 poli e nelle librerie CoDeSys ho pensato di inserire le funzioni IEC per la gestione diretta di tale porta. Infatti è da considerare che tutte le librerie IEC possono essere create con lo stesso linguaggio IEC (da chi programma l'applicativo) oppure possono far riferimento diretto a funzioni sviluppate in linguaggio C ed integrate, dopo la loro compilazione, nel programma RTS.
L'implementazione delle funzioni di lettura ingressi e scrittura uscite è solo una delle cose da sviluppare per adattare CoDeSys allo specifico hardware. Infatti possono esserci altre risorse e dispositivi da gestire nel RTS (ad esempio un orologio, una porta di comunicazione o delle specifiche funzioni di libreria del linguaggio IEC). Inoltre occorre definire la strategia, ed implementarne il relativo software, per il **salvataggio delle memorie ritentive** quando la scheda viene spenta. La creazione di una serie di funzioni di risposta, specifiche dell'hardware e chiamate dall'ambiente CoDeSys, permette poi di realizzare il **PLC browser** ossia una sorta di terminale integrato nell'ambiente di sviluppo utile per la configurazione e la gestione di servizio del PLC.
L'argomento ha una certa complessità e richiederebbe altro spazio e pazienza del lettore, ma alla fine ciò che conta è che il **programma RTS** assieme ad altri files da installare sulla scheda, una volta per tutte, è pronto e disponibile a questo link:

Download

EY-CPU_Software_PLC.zip

Installazione del software RTS sul PLC

La scheda per ora non contiene alcun particolare software se non il sistema operativo preinstallato sui processori nuovi di stecca. Per fare un esempio con una situazione analoga è come se avessimo appena acquistato un Personal Computer con preinstallato esclusivamente Windows. Questo PC però potrebbe disporre di una versione del sistema operativo non perfettamente aggiornata e quindi, per essere sempre al passo coi tempi, provvederemo, per prima cosa, ad un suo aggiornamento. Il passo successivo sarà poi quello di installarci uno o più programmi applicativi per fare qualche cosa di specifico di nostro gradimento (un gioco, un programma di scrittura o altro). Nel caso invece della nostra scheda installeremo il programma RTS che la trasformerà in un PLC programmabile dal tool CoDeSys.
Per fare tutte le operazioni di servizio ed installazione software si utilizza una connessione mediante la porta Ethernet tra la scheda ed un PC. A dire il vero esistono due possibilità di connettere la scheda al PC tramite la porta Ethernet. La prima è quella di utilizzare un **cavo diretto** tra il PLC e la porta del PC. La seconda è quella di connettere il **PLC ad un router** di una rete locale. Nel primo caso, apparentemente il più semplice, occorre usare un cavo Ethernet incrociato (dove le connessioni di trasmissione e ricezione dati sono mutuamente incrociate tra i due apparecchi). Questa tipologia di cavo è in genere evidenziata da un scritta, tipo CROSS o altro, oppure dal colore rosso. Nel secondo caso, apparentemente il più difficile, si tratta invece di usare un cavo non incrociato (ed anche molto più comune) tra il PLC ed un router.
Alcuni ora penseranno: ma dove lo vado a prendere adesso un router? Eppure con la grande diffusione di Internet nelle case e nei luoghi di lavoro, penso che ci saranno più router che portaombrelli. Infatti quasi tutti i modem ADSL sono anche dei router. Già da tempo si stanno realizzando, anche inconsapevolmente, reti locali in casa. Basti pensare che, oltre alla connessione in rete di più PC col Wi-Fi o via cavo, molte altre apparecchiature, come sistemi di archiviazione multimediale, televisori LCD e console di videogioco, sono connessi, tra loro e verso il mondo esterno Internet, tramite un router. Tra poco anche il nostro PLC farà parte di questa grande famiglia. E' per questo che la seconda modalità di connettere il PLC ad un PC, ossia tramite un router, è estremamente più utile e funzionale, nonchè alla fine più facile.
Prendiamo in esame un classico modem-router ADSL, ormai acquistabile con pochissime decine di Euro. Questo apparecchi in genere hanno 4 **porte Ethernet**, un'antenna **Wi-Fi** ed una connessione **ADSL** alla rete telefonica. I dispositivi connessi alle 4 porte via cavo, più tutti quelli trovati in Wi-Fi nell'immediate vicinanze, più il modem ADSL, fanno parte di un'unica rete locale di apparecchi che possono scambiarsi reciprocamente dei pacchetti di dati qualsiasi e per scopi diversi. Ognuno degli apparecchi connessi alla rete può decidere di inviare dei pacchetti dati ad un qualsiasi altro apparecchio della rete.
Arrivato a questo punto, prima di proseguire, voglio inserire un paragrafo con

informazioni di carattere generale riguardo le reti Ethernet ed il loro funzionamento, parentesi fondamentale per la comprensione delle parti successive del mio blog.

Nozioni elementari sulla rete Ethernet

Voglio riportare il classico esempio, che si trova ampiamente in Internet sotto varie forme, che meglio descrive il funzionamento di una rete Ethernet. Immaginiamo di dover spedire dei documenti che abbiamo racchiuso in un pacchetto postale. Per far questo ci attacchiamo sopra un'etichetta con il nostro indirizzo di mittente e, ovviamente, l'indirizzo del destinatario. A questo punto ci rechiamo col pacchetto all'ufficio postale della nostra città per spedirlo. Appena entrati troviamo subito due file a due sportelli distinti: uno per le **spedizioni locali** nella propria città e uno per le **spedizioni fuori città**. A questo punto dobbiamo fare la scelta corretta (per non fare la fila due volte!) e per questo controlliamo, per sicurezza, gli indirizzi del mittente e destinatario che abbiamo scritto sul pacchetto. Isoliamo in particolare i campi dell'indirizzo corrispondenti alla **nazione** e alla **città**. A questo punto confrontiamo solo queste informazioni tra il mittente ed il destinatario e decidiamo quale fila scegliere. Per sicurezza abbiamo controllato la parte di indirizzo partendo anche dalla nazione per evitare di sbagliare nel caso esista una città con lo stesso nome ma in due nazioni differenti.
Se, dopo la scelta corretta, decidiamo di consegnare il pacchetto all'impiegato della fila relativa alla propria città, questo chiama il suo postino di fiducia che, in bicicletta, correrà subito a consegnarlo al destinatario. Se invece il pacchetto era destinato fuori città, l'impiegato addetto alla corrispondente fila prende il pacchetto e lo cede al suo aiutante, il quale lo porterà alla stazione ferroviaria della città. Da qui il primo treno lo prenderà in consegna e lo porterà via dalla città del mittente. Ora il percorso del nostro pacchetto potrebbe essere più o meno lungo e complesso e potrebbe addirittura coinvolgere più mezzi, come altri treni o aerei. Al termine del lungo viaggio il pacchetto arriverà di nuovo in un ufficio postale, quello della città del destinatario. A questo punto l'impiegato dell'ufficio postale chiamerà il suo postino e lo farà consegnare finalmente al destinatario. Il destinatario poi apre il pacco, legge i documenti e decide di rispondere inviando a sua volta un nuovo pacchetto. E così si ripete il giro di nuovo nel senso inverso.
Ho voluto descrivere per primo l'esempio completo di cosa avviene in un caso molto comune a tutti. Ora, con qualche analogia, descriverò ciò che avviente in una rete di computer o altri apparecchi. I vari dispositivi collegati alla rete, ognuno con un proprio indirizzo (l'**indirizzo IP**) sono gli utenti che preparano i pacchetti e ci scrivono sopra l'indirizzo del mittente e destinatario. Questi pacchetti arrivano al **router** (l'uffico postale della propria città) tramite il cavo (la strada tra casa e l'ufficio postale) oppure tramite la connessione Wi-Fi (cioè saltando nell'aria senza toccare la strada). Il processore del mittente, analizzando alcuni campi dell'indirizzo (secondo la **Netmask** o maschera di selezione dei soli campi utili per capire se la città è la stessa o meno) decide se inviare il pacchetto ad un utente della stessa rete locale (la

propria città) oppure se consegnarlo allo sportello abbinato alla stazione ferroviaria sita all'indirizzo del **Gateway**. Questo indirizzo è quello di un particolare destinatario intermedio, ma ancora facente parte della rete locale della città, che ha l'accesso al mondo esterno. Questo dispositivo coincide con il **modem ADSL** che è spesso inscatolato assieme al router dentro l'apparecchio che siamo soliti vedere. In questo modo il pacchetto inizia la sua avventura in giro per il mondo, passando e saltando attraverso stazioni intermedie, ponti radio, centrali ed altre apparecchiature. Infine il pacchetto arriva al router del destinatario (il secondo ufficio postale) per essere finalmente consegnato al dispositivo di arrivo, in genere un server Web, il quale risponderà con l'invio dei suoi pacchetti. Questo è quanto avviene quando si naviga in Internet ed è anche alla base della funzione di Webserver del PLC didattico che vedremo in futuro.

Voglio infine fare un breve accenno ad una funzione di servizio molto importante svolta dal router. Per un corretto invio dei pacchetti di comunicazione è fondamentale che ogni dispositivo di una rete locale abbia un indirizzo IP univoco e distinto dagli altri. Per questo un metodo molto semplice è di rivolgersi al router, che ha una visione completa di tutti i dispositivi connessi e dei loro indirizzi, per chiedergli la cortesia di assegnarcene uno ancora disponibile. Questo servizio si basa sul protocollo Dynamic Host Configuration Protocol, abbreviato in **DHCP**, ed è disponibile sui router ed i server di rete. Questo indirizzo non è tuttavia definitivo, ma dinamico, perchè ha una scadenza (ad esempio 1 giorno) e quindi in momenti differenti l'indirizzo assegnato può cambiare. L'alternativa all'assegnazione automatica è l'assegnazione manuale da parte dello stesso dispositivo della rete. Questo però richiede di aver chiara la situazione, a livello di indirizzi, di eventuali altri dispositivi ad assegnazione manuale, per evitare di sceglierne di già utilizzati.

L'utility Chiptool per la connessione al PLC

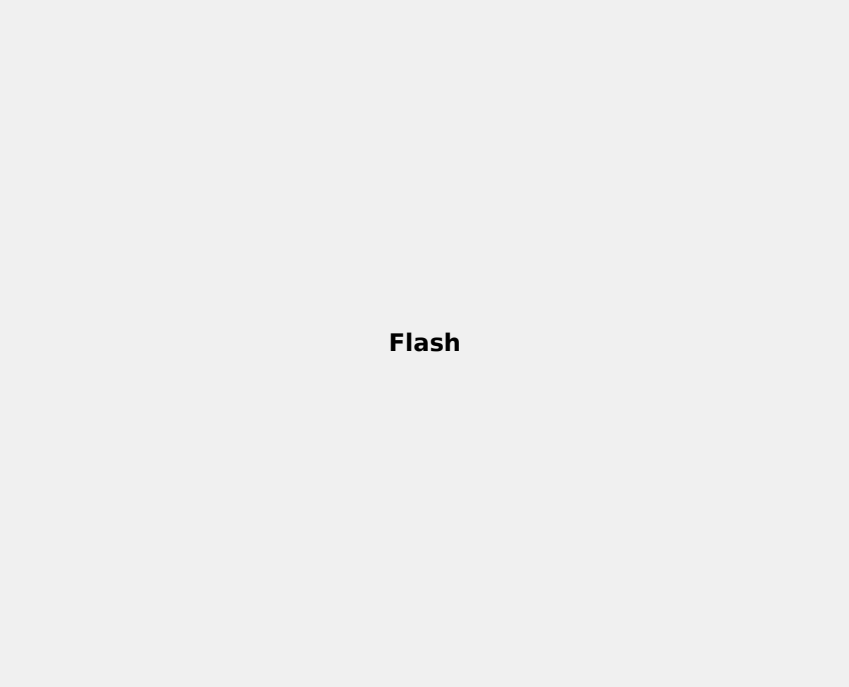
Finalmente, dopo le opportune informazioni di base, procediamo all'aggiornamento del sistema operativo e all'installazione sulla scheda del software RTS e di altri files, il cui download è disponibile al link precedente. Questa fase è necessaria solo la prima volta, dopo la realizzazione dell'hardware. Per il successivo utilizzo come PLC tutte le operazioni di programmazione saranno poi svolte in modo autonomo dal solo ambiente di sviluppo CoDeSys. Questa fase di preinstallazione dei files sulla scheda richiede l'utilizzo di un software di utility, il **Chiptool**.
Il file di setup di tale utility è il seguente:

Download

Chiptool.zip

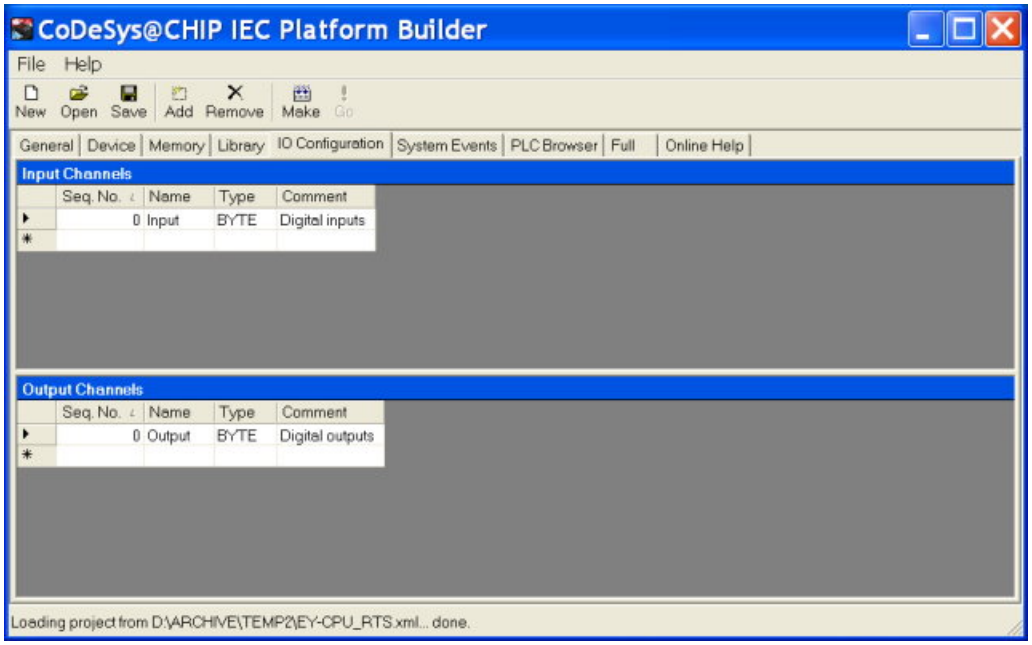
Per installare l'utility basta eseguire il file di setup dopo averlo unzippato. Una volta

lanciato il programma su un PC connesso nella stessa rete locale dove è inserita anche la scheda PLC, tale utility inizierà una continua scansione della rete alla ricerca di tutti i dispositivi, indipendentemente dalla loro attuale configurazione di rete. Il Chiptool è un collezione di semplici utility per realizzare varie operazioni di servizio sul processore. Per ora ci limiteremo ad utilizzare solo quelle fondamentali allo scopo di concludere, finalmente, la preparazione del nostro PLC.
Ho pensato che il modo migliore, ed anche più compatto, per spiegare le operazioni che ora dovremo effettuare sulla scheda, sia quello di raccontarvelo con un **video** fatto al meglio delle mie possibilità, praticamente da 4 in pagella...
Dimenticavo: per vedere qualcosa di comprensibile consiglio di impostare la risoluzione su 720p (HD) e di metterlo a schermo intero.



Creazione dei files di supporto a CoDeSys

L'ambiente di sviluppo CoDeSys richiede, per operare con un specifico PLC, l'intallazione del Target (sinonimo di generico apparecchio da programmare). Questo avviene mediante l'aggiunta di un insieme di files, chiamati **TSP** (Target Support Package), nella cartella Targets presente sotto l'istallazione di CoDeSys (in genere C:\Programmi\3S Software\CoDeSys V2.3\Targets).
I files del TSP contengono tutte le informazioni specifiche dell'hardware utilizzato, in modo che il tool di programmazione possa adattarsi a questo. Per la creazione di questo KIT di files si utilizza un'utility software che genera automaticamente l'intero TSP. Voglio solo mostare, come esempio, una delle pagine di impostazione della configurazione degli **Ingressi/Uscite** del PLC.



Utilizzo dell'utility per la creazione del TSP

Nel caso della scheda in questione ho definito una variabile di tipo **BYTE** per gli **8 ingressi digitali** ed una variabile, anch'essa di tipo **BYTE**, per le **8 uscite digitali**. Lo scopo dell'utility, riguardo a tali informazioni, è di generare un file di configurazione che permetterà a CoDeSys di considerare la presenza di queste risorse nelle aree di memoria immagine degli IO. Inoltre verranno dichiarati i due simboli **Input** ed **Output** (utilizzabili nel programma IEC come variabili globali) relativi agli indirizzi effettivi di questi due bytes. Sono proprio questi due bytes di memoria ram quelli che abbiamo gestito mediante le due funzioni create con lo sviluppo del RTS della scheda, legandoli allo stato degli optoisolatori di ingresso e dei relè di uscita.
Vi risparmio ulteriori particolari su tale fase di preparazione, che tra l'altro non ha nulla di particolamente interessante. Il **TSP pronto all'uso** per il PLC didattico è al seguente link, pronto per essere installato successivamente all'installazione di CoDeSys:

Download

EY-CPU_Software_TSP.zip

Installazione del software CoDeSys su PC

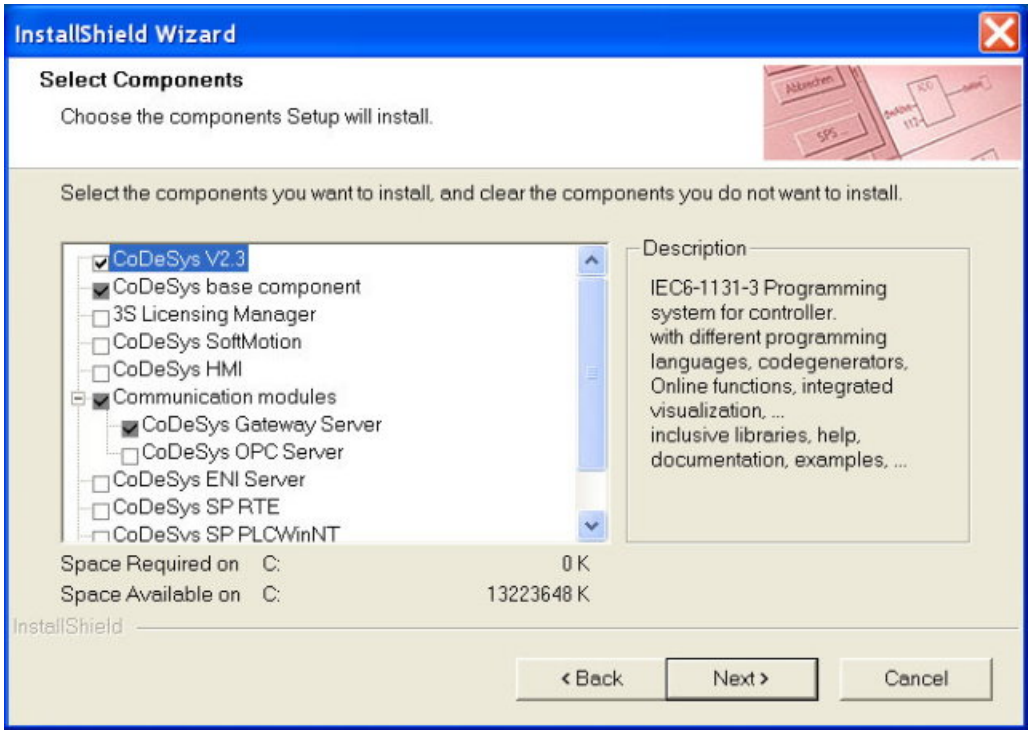
Ora installeremo sul PC l'ambiente di sviluppo CoDeSys. Il software è piuttosto complesso in quanto rivolto alla programmazione IEC di veri PLC, ben più grandi e potenti del mio piccolo PLC didattico. Infatti circa 250 produttori mondiali utilizzano

questo tool per la programmazione dei loro sistemi. Tuttavia basta, inizialmente, non guardare tutti i menu e le numerose opzioni offerte e cominciare subito a scrivere semplici programmi, magari seguendo le mie indicazioni di minima o utilizzando il mio programma di **template** preconfigurato. L'ambiente contiene manuali dettagliati ed help in linea che, non solo spiegano il funzionamento del tool, ma forniscono anche numerosi esempi di programmazione IEC. Questo potente tool è disponibile per il download anche dal sito della 3S, la casa produttrice del software, ma preferisco per completezza inserire il seguente link:

Download

CoDeSys.zip

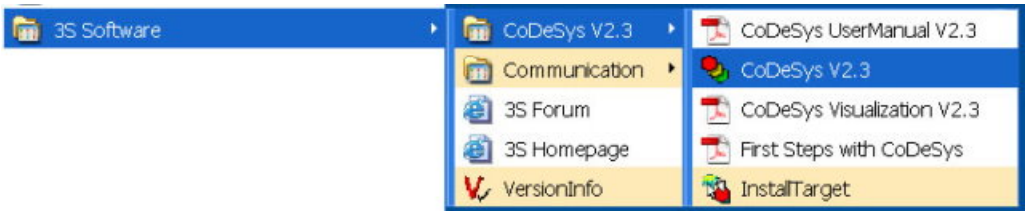
Dopo averlo unzippato, lanciamo il file di **setup di CoDeSys**, confermando le solite schermate di installazione fino ad arrivare alla seguente richiesta di configurazione delle opzioni:



Selezione dei tools da installare

Il pacchetto software comprende alcune opzioni delle quali alcune richiedono un'ulteriore specifica licenza. Queste opzioni sono funzioni molto particolari e sofisticate che vanno ben oltre gli scopi, non solo di un PLC didattico, ma anche di un PLC da usare in una complessa macchina. Infatti la licenza base permette l'utilizzo completo di tutte le funzionalità di programmazione IEC e di espansione mediante

CANopen. Quindi per continuare l'installazione impostare le sole selezioni riportate nell'immagine precedente.
Al termine dell'installazione, tramite il menu Start di Windows, saranno disponibili anche i principali **manuali in formato PDF**.



Cartella del menu Start con i manuali PDF

Altra documentazione è reperibile sotto la **cartella Documents** dell'installazione stessa.
L'installazione standard di CoDeSys richiede successivamente l'impostazione di alcuni registri di sistema del PC per configurare la connessione remota, da rete Internet, della scheda con il tool. Inoltre occorre aggiungere un parametro nella pagina html base utilizzata per creare le pagine del Webserver. Senza entrare nei dettagli ho preparato allo scopo un semplice file eseguibile in batch accompagnato da alcuni piccoli files di supporto alle operazioni:

Download

EY-CPU_CoDeSys_config.zip

Dopo aver unzippato il file in una cartella del PC, basta eseguire il file **CoDeSys_config.bat** per configurare queste poche cose. Infine nello zip ho inserito anche il file **usbserd.inf** che è il driver USB da installare nel caso si utilizzi tale porta come seriale COM virtuale per la programmazione CoDeSys. Comunque il consiglio è di usare la porta USB della scheda per realizzare un secondo hard-disk con una chiavetta di memoria da qualche Gigabytes e di programmare tramite la porta Ethernet.
Per concludere l'installazione del software su PC occorre fornire a CoDeSys l'insieme dei files relativi al TSP (Target Support Package) che identificano le caratteristiche e risorse dello specifico Target (la scheda EY-CPU). Per questo occorre unzippare il file EY-CPU_Software_TSP.zip, il cui link è al precedente paragrafo, in una cartella del PC e poi eseguire il file batch **EY-CPU_install.bat**. Confermare poi l'installazione ed attendere la chiusura della finestra di esecuzione. A questo punto tutti i files del TSP sono disponibili nella cartella Target di installazione di CoDeSys. Il tool ora riconoscerà solo la EY-CPU come PLC utilizzabile, ma la flessibilità di CoDeSys permette l'installazione di tanti TSP relativi ad altrettanti diversi PLC o schede per l'automazione.

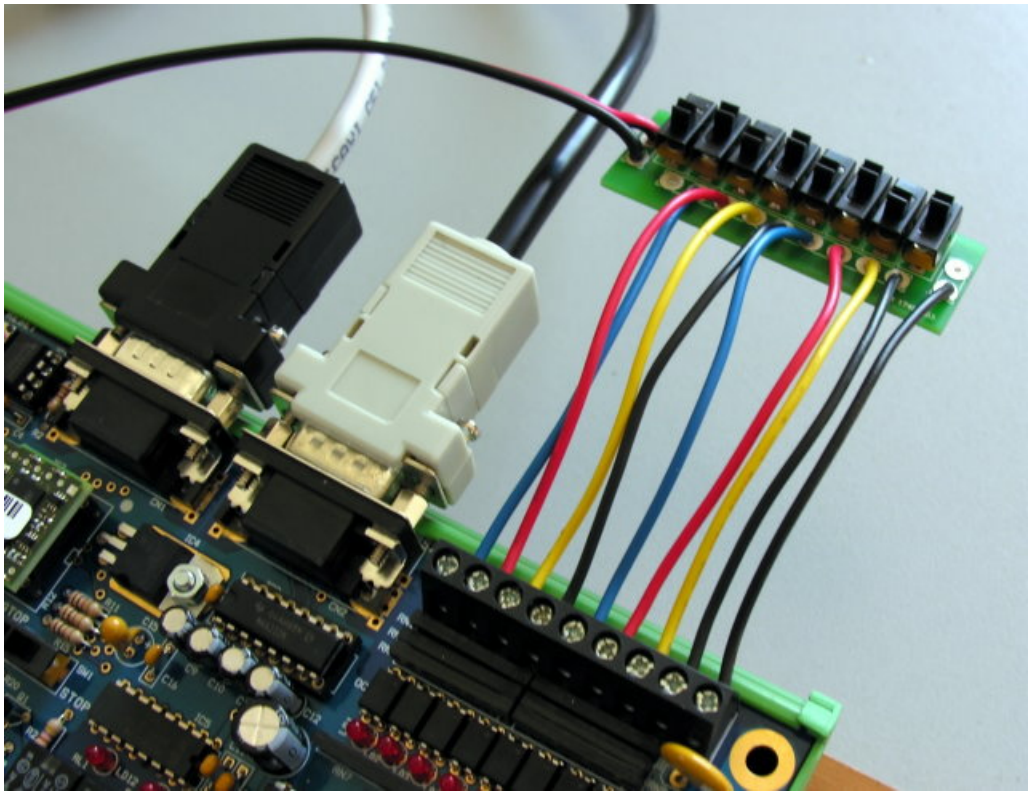
Finalmente proviamo la scheda PLC!

Dopo tanto lavoro finalmente è tutto pronto per la prova finale e non posso attendere oltre per dare corrente e provare la scheda. Nel prossimo blog farò uso del PLC per proporre esempi di programmazione IEC, cercando di essere più semplice e didattico possibile. Per questo tralascierò ogni dettaglio di contorno dell'ambiente di programmazione per concentrarmi solo su semplici e corti listati di programma IEC. In più ho voluto pensare a tutti quanti hanno avuto la pazienza di seguirmi e non soltanto a chi si è voluto imbarcare, come me, nella costruzione dei pochi esemplari di KIT dei quali ho radunato i pezzi. Per questo ho realizzato un primo programma CoDeSys che costituirà una base di partenza come template per un nuovo progetto IEC e permetterà a chiunque di mettere in pratica, sia i miei esperimenti ed esercizi di programmazione, sia i propri. Ecco il link al progetto CoDeSys che ha lo scopo di template:

Download

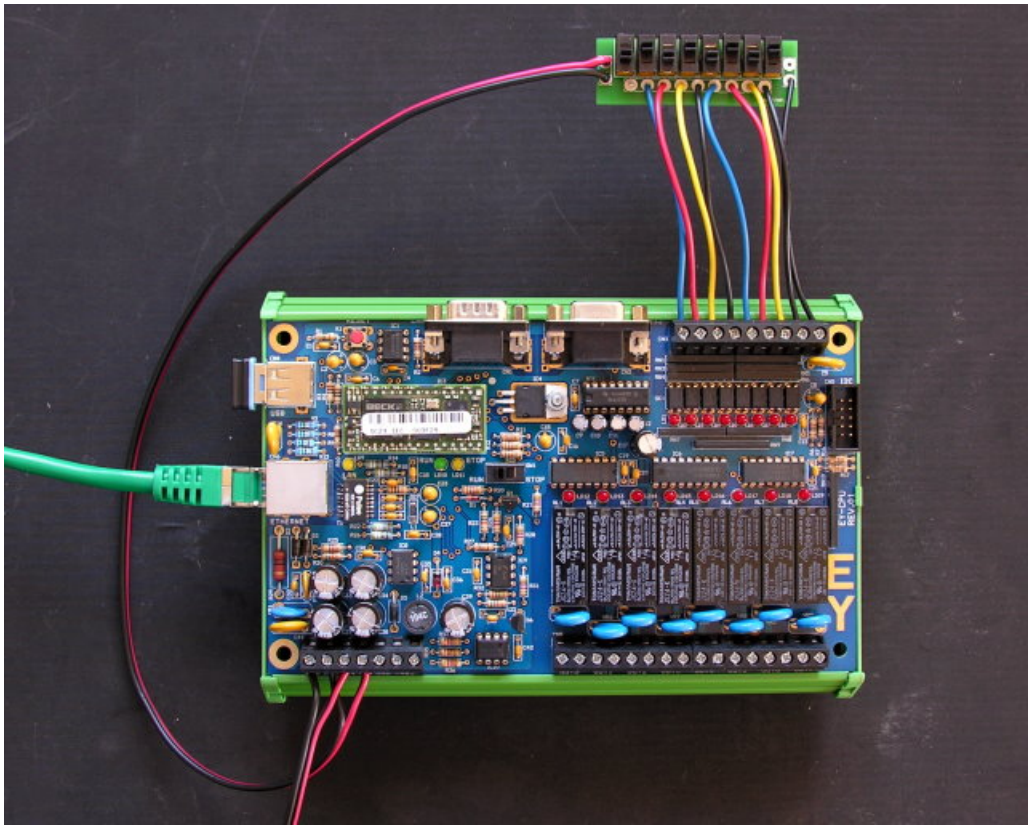
EY-CPU_Template.zip

Il programma di **template** comprende già tutte le impostazioni di partenza di un nuovo progetto, comunque semplici, ma che per il momento voglio evitare per andare direttamente al cuore della programmazione. Inoltre nel template ho preparato una finestra di **Visualizzazione grafica** creata con una foto reale della mia scheda alla quale ho sovrapposto oggetti collegati a tutti i gli ingressi ed uscite digitali. In questo modo chiunque potrà provare e fare pratica con il PLC didattico come se lo avesse realmente costruito. Chi utilizzerà la scheda reale dovrà applicare dei veri segnali ai morsetti, mentre chi utilizzerà la **versione virtuale del PLC** attiverà gli ingressi facendo click con il mouse sugli ingressi del PLC virtuale.
In un PLC didattico reale non può mancare ovviamente il simulatore degli ingressi digitali a levette:



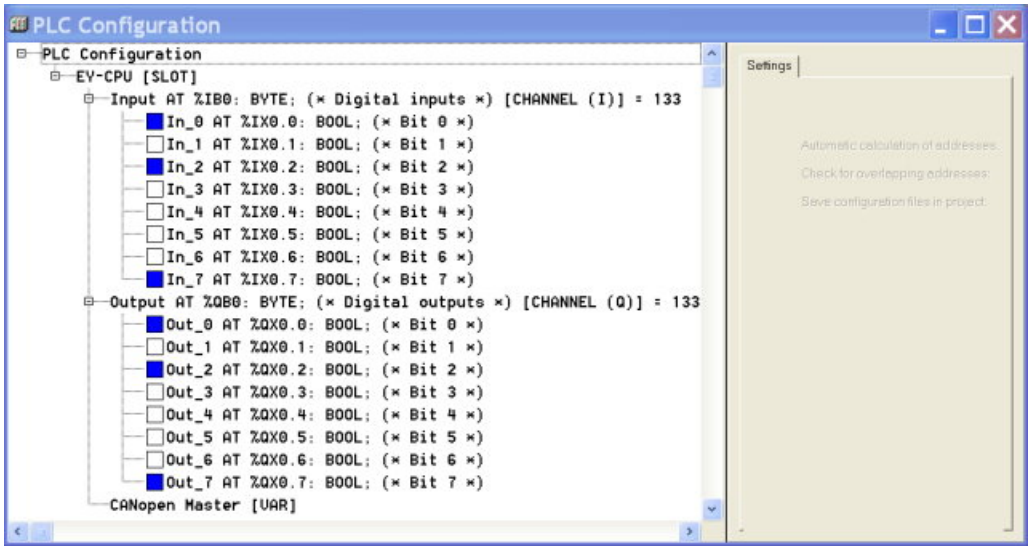
Il simulatore a levette degli ingressi digitali

Non dimentichiamoci anche di fare tutti gli altri collegamenti alla scheda o almeno l'alimentazione a 24V in basso a destra e l'Ethernet sul lato sinistro:



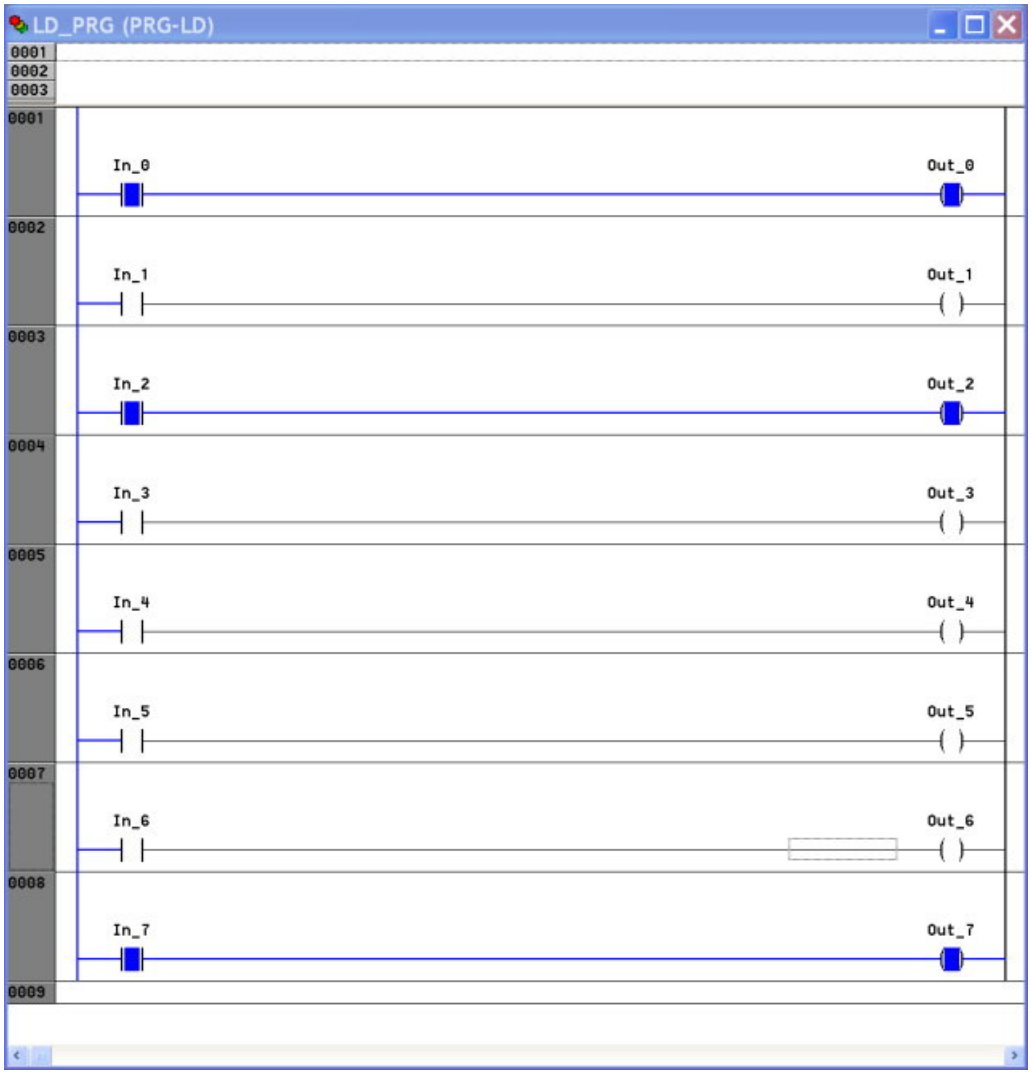
Le connessioni per la prova della scheda

Per fortuna la finestra del PLC Configuration dell'ambiente di programmazione ci conferma che, almeno gli ingressi ed uscite digitali, funzionano correttamente:



La finestra di riferimento e mappatura degli IO

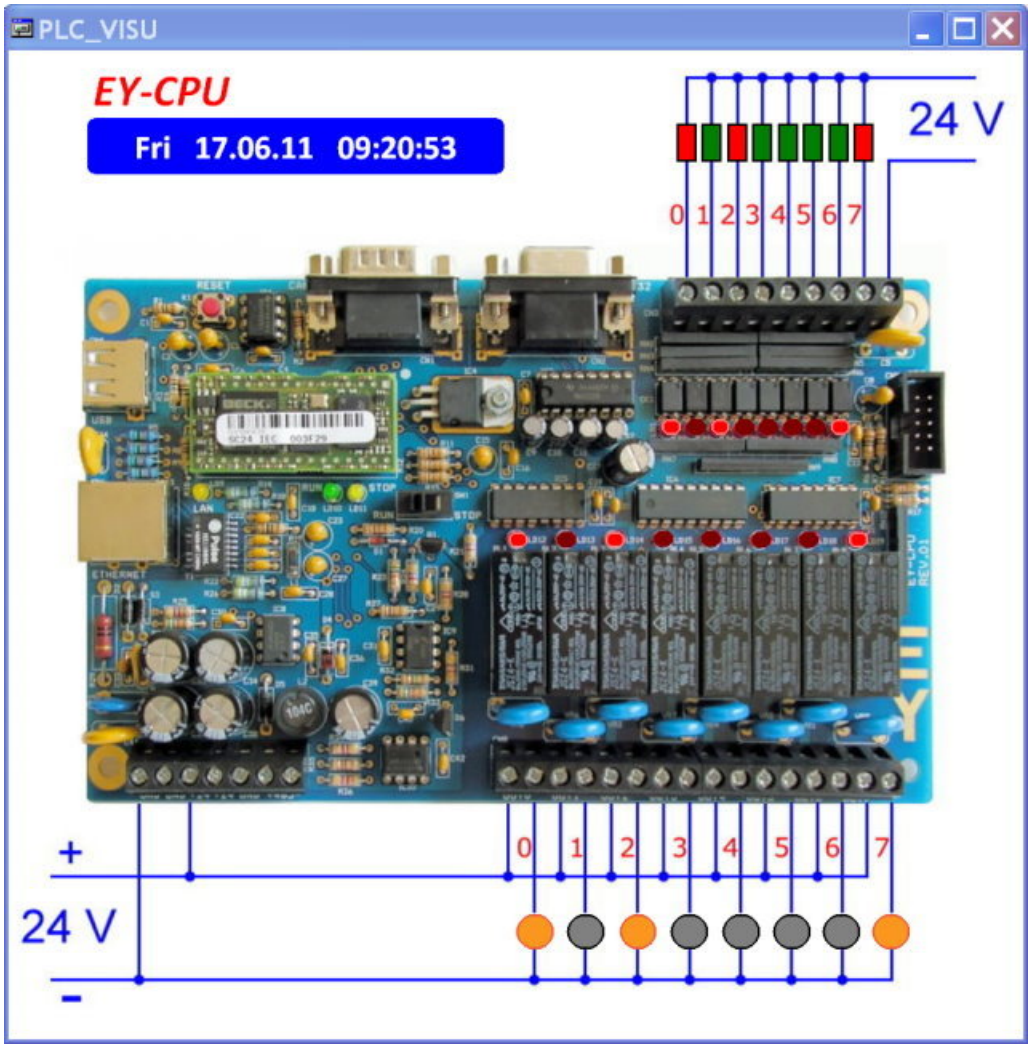
Come si vede le risorse di IO, subito disponibili, sono i due bytes chiamati Input ed Output che avevamo dichiarato nella creazione del TSP. Inoltre ho predisposto la visibilità diretta anche dei singoli bits del byte, associandogli un nome convenzionale (In_0...In_7 e Out_0...Out_7) così da inserirli direttamente nel listato del programma applicativo come variabili booleane. Le label identificative scelte sono comunque cambiabili in tale finestra per dargli dei nomi più facili da ricordare, come **Accendi** e **Spegni** per gli ingressi e **Lampada** per un'uscita. Per testare gli ingressi e le uscite ho inserito nel programma template anche una POU in **linguaggio Ladder** che collega singolarmente ogni ingresso con un'uscita (un classico immancabile!):



Programma Ladder per test ingressi/uscite

Ora il PLC reale funziona ed è pronto per fare dei giochi su un bel tavolo del laboratorio, mostrando orgoglioso tutte le sue capacità. Purtoppo per lui, al termine dei miei blog, gli dovrò fare la brutta sorpresa di costringerlo al lavoro, rinchiudendolo in un buio garage, per controllare le luci del giardino (così si

guadagna la pagnotta anche lui!). Ma per ora non diciamogli niente, altrimenti si condiziona e poi mi fa fare delle brutte figure...
Come ho già anticipato, ho inserito nel template anche una finestra di visualizzazione simulata del mio PLC reale:



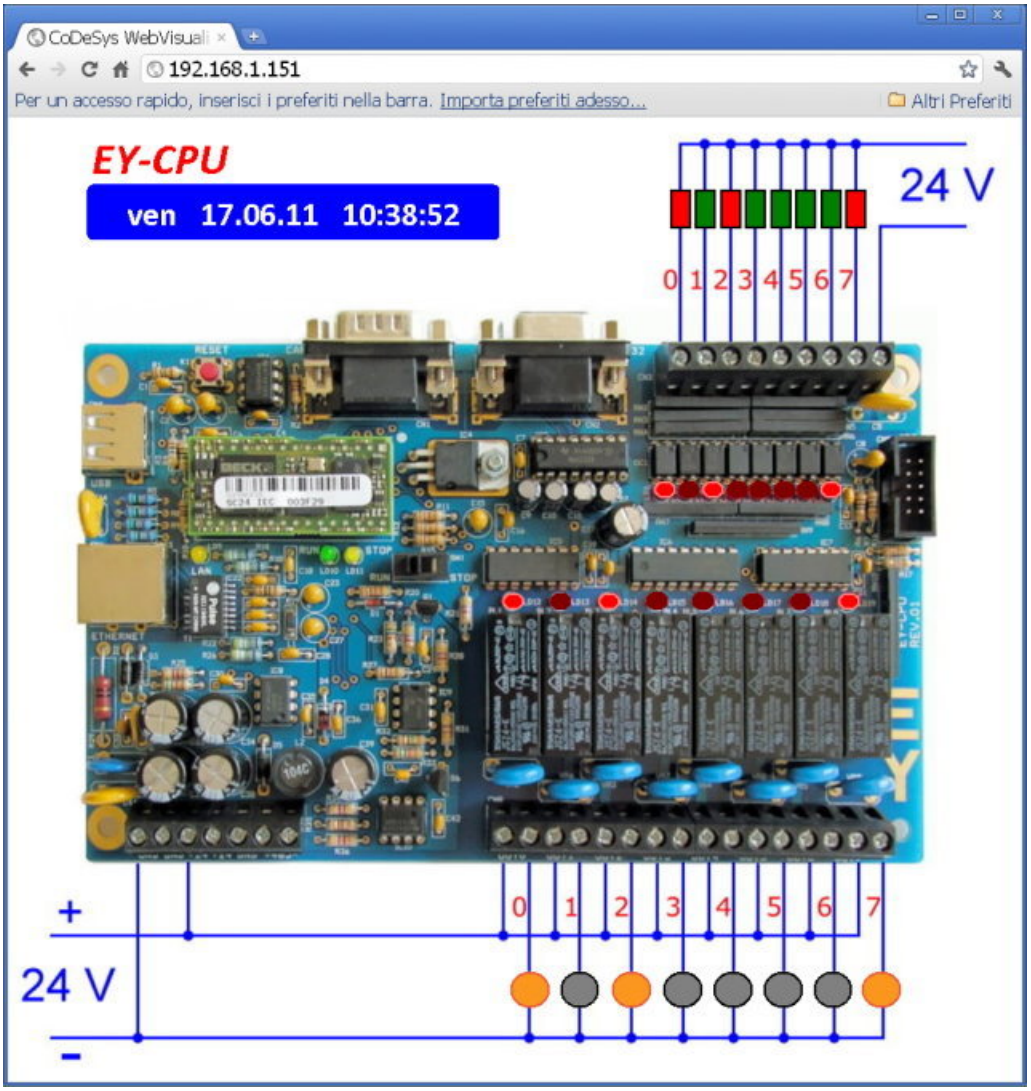
Visualizzazione della versione virtuale del PLC didattico

Questa finestra è stata creata all'interno dell'ambiente di sviluppo utilizzando degli **oggetti grafici** piazzati su un'immagine di sfondo che ho preparato preventivamente. Questi oggetti hanno associate delle **proprietà** che possono essere modificate, in tempo reale, tramite le variabili utilizzate nel programma. Per esempio, per far accendere i led sulla scheda, ho posizionato dei semplici cerchi dove il colore interno commuta tra due colori a seconda del valore del relativo bit di ingresso o uscita. Per il simulatore degli ingressi ho utilizzato degli interruttori rettangolari che cambiano di colore (verde spento, rosso acceso) facendo click su questi col mouse.

Il mio programma base può essere utilizzato in vari modi a seconda anche che si disponga o meno del PLC reale:

- Si dispone materialmente della scheda e si esegue il programma reale su questa. In tal caso l'immagine grafica, sul PC connesso, del PLC virtuale riporta esattamente lo stato degli ingressi/uscite correnti operando solo come monitor.
- Si dispone materialmente della scheda ma si va in simulazione software del programma col PC. In tal caso il PLC virtuale della finestra, non solo visualizza lo stato degli ingressi ed uscite, ma controlla in modo attivo il programma tramite i simulatori virtuali degli ingressi. Il PLC reale è invece momentaneamente inattivo.
- Non si dispone del PLC reale. Allora è possibile lavorare solo in simulazione software del programma utilizzando il PLC virtuale, sia per monitorare lo stato degli IO, che per gestire il programma tramite i simulatori virtuali degli ingressi.

La stessa finestra di visualizzazione del PLC virtuale è anche subito utilizzabile all'esterno dell'ambiente di programmazione, in totale autonomia da questo. Basta solo collegarsi con un **Browser** all'indirizzo della rete locale associato al PLC e fare click con il mouse sugli ingressi. Ovviamente questo può avvenire anche dall'altra parte del mondo, ma la funzione di Webserver richiederebbe uno spazio apposito. Infatti dovrei continuare a spiegare ancora alcune cose riguardo le reti Ethernet ed i router. Per ora mi sono semplicemente collegato alla rete locale, all'indirizzo 192.168.1.151, utilizzando il browser **Google Chrome**:



Il PLC virtuale visto dalla rete Ethernet

Conclusione

Termina qui la realizzazione del PLC didattico con tutto quello che riguarda il progetto e la sua messa in funzione. Ormai sono agli sgoccioli con i miei KIT rimediati, ma sono contento di vedere, a breve, anche loro accesi. Almeno il mio PLC, quando controllerà le luci del giardino, saprà che ci sono dei fratelli sparsi in altre città che fanno un mestiere simile. E chissà se riusciranno un giorno a mettersi in contatto Internet, con le librerie TCP/IP, per finalmente conoscersi e vincere la paura del buio del garage.
A parte gli scherzi e i saldatori, la sola creazione del target EY-CPU, con i relativi files associati, è anche un pretesto per mettere in opera il tool CoDeSys su un **economicissimo PLC virtuale**, che poco differisce da quello reale visto lo scopo didattico che mi ero proposto. Penso che già questo possa costituire un valido

strumento educativo, utile per fare prove ed esercizi su quello che è il linguaggio di programmazione dell'automazione.

Estratto da "http://www.electroyou.it/mediawiki/index.php?title=UsersPages:Aox:software-del-kit-iec61131-3"