AOX

BLOCCHI FUNZIONE PER LA DOMOTICA

6 August 2011

Premessa

Durante il progetto e la realizzazione del mio **PLC in KIT** mi sono sempre chiesto che cosa poi ne avrei fatto una volta terminato. Sapevo solo che avrei dovuto utilizzarlo per parlare un po' di **programmazione IEC61131-3**, ma senza avere un'idea precisa su come affrontare l'argomento. Dato che però ero concentrato sull'**hardware** e sul **sistema operativo**, le cose che mi piace maggiormente fare, non ho fatto molta fatica a non pensarci e quindi ho sempre rimandato questa decisione. Così ho preso tempo per ben **tre blog** fino a quando, illuminato dalla lettura del **Forum di EY**, ho deciso di utilizzarlo per fare degli esempi concreti di programmazione dedicati alla **domotica**. Quindi ho messo insieme questi ingredienti: una passata esperienza di progettazione di apparecchiature da installazione civile, un PLC nuovo di zecca in cerca di lavoro e la mia intenzione di affrontare la programmazione IEC. Per questo ho pensato di fare alcuni esempi concreti di software IEC con lo scopo di realizzare dei **componenti da installazione civile** e di collaudarli sul mio nuovo PLC. Al termine riunirò questi oggetti, dalla funzione ben definita, in un libreria domotica. Da questo magazzino di apparecchiature, un giorno attingerò una quantità a piacere di questi piccoli, ma comodi, componenti per realizzare un **quadretto elettrico virtuale** per la casa.

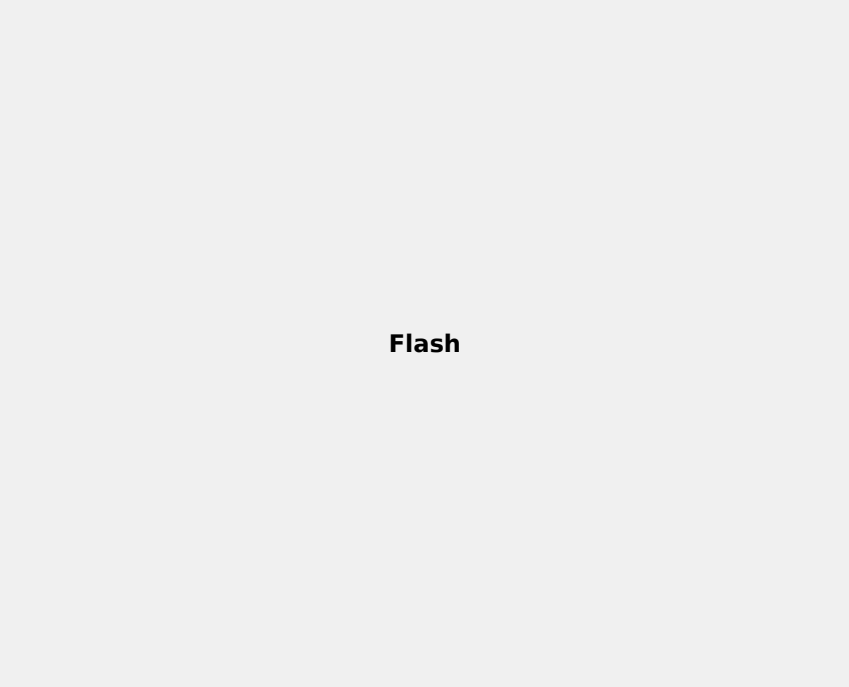
Un breve punto della situazione

Finalmente il **PLC è pronto** e funzionante, l'ambiente di sviluppo **CoDesys è installato** e quindi non rimane che procedere al suo utilizzo. Voi direte quanta fatica fatta prima di arrivare ad usare un PLC, con tutti quelli che ci sono già pronti!!! D'altra parte, il tutto è nato dal mio primissimo articolo **Elettronica nell'automazione industriale**: se nel realizzare una **scheda elettronica dedicata** la si fa pure **programmabile come un PLC**, tanto meglio, possiamo sempre scegliere come programmarla!
Per chi non avesse seguito i precedenti 3 blog della serie **un KIT programmabile IEC61131-3**, dei quali consiglio una paziente e propedeutica lettura, ricordo che esiste una **versione immateriale del PLC**, ossia simulata dal software del PC. Alcuni volonterosi, oltre me, hanno voluto **scottarsi col saldatore** per realizzarne anche uno in **KIT**, ma quello che proporrò nel seguito prescinderà dalla presenza di un **PLC materiale** e gli esempi riportati potranno essere provati e modificati da

chiunque. Chi vuole inoltre sperimentare lo standard IEC, anche a livelli approfonditi, creando da zero dei propri programmi, potrà poi testarli sul mio PLC virtuale: basta solo un PC e un po' di buona volontà.

Piccolo esercizio di riscaldamento

Nel precedente blog avevo tirato piuttosto diritto sull'utilizzo dell'**ambiente di programmazione CoDeSys** e così farò anche nel seguito. Avevo fatto vedere alcune immagini del **programma di prova**, contenente le istruzioni per il test degli IO, con già preimpostate alcune cose necessarie quando si inizia un progetto. Questo programma può essere usato come base di partenza di un nuovo progetto tramite il menu **File -> New from Template**, così tutto quello che serve per iniziare è già configurato. Infatti, se non si parte da tale programma base, la prima cosa in assoluto da fare, quando si crea un nuovo progetto, è selezionare il modello di PLC, detto **target**. L'imbarazzo della scelta è grande quando nell'elenco a tendina compare solo il **PLC "none"** e il **PLC "EY-CPU**" e, chissà perchè, ho proprio scelto "EY-CPU". Di conseguenza alla selezione del PLC, viene presentata un finestra di impostazione delle **opzioni del target**. Inoltre, visto che il PLC è predisposto per la gestione del **CANopen**, occorre anche caricare il modulo di comunicazione MASTER. Sono tutte operazioni semplici e documentate nei manuali, ma, partendo dal mio template, sono già tutte predisposte.
Nel programma di template sono però presenti anche due oggetti in più. Il primo è un modulo di programma (detto POU) in linguaggio ladder che collega gli 8 ingressi digitali della scheda con le 8 uscite a relè per il **test degli IO**. Questo modulo, in un nuovo progetto, deve essere ovviamente eliminato dall'elenco degli oggetti. Il secondo è una finestra di **visualizzazione grafica**, con la foto della scheda reale e gli oggetti colorati dinamicamente per simulare lo stato degli IO. Questo form grafico può essere anche mantenuto nel programma da sviluppare in quanto non dà nessun fastidio e comunque fornisce una rappresentazione visiva del PLC con funzione di monitor ed attivazione degli IO.
Ho preparato un **semplice e rapido video** che spiega come iniziare a fare un nuovo programma, tanto per scaldare il motore del nostro PLC. Il video andrebbe guardato a schermo intero, impostando la risoluzione su 720p (HD):



Cos'è un blocco funzione

Il concetto di blocco funzione è di estrema importanza nel linguaggio standard IEC61131-3. Per questo voglio dedicargli particolare attenzione iniziando a fare esempi di programmazione proprio creando alcuni blocchi funzione, ciascuno con una sua pratica applicazione.
Ricordiamo che il PLC era stato ideato per sostituire complessi **quadri elettromeccanici** con un qualcosa di più flessibile, grazie alla simulazione software di oggetti fisici e componenti reali come i **relè**. E' dall'estensione di tale concetto iniziale che nasce il blocco funzione come **generico componente** materiale (uno scatolotto con le morsettiere) che svolge una funzione predefinita, più o meno complessa. Possono essere esempi di blocco funzione un **temporizzatore**, un **termoregolatore** o un **programmatore orario**. Questi sono tutti "pezzi" materiali che un elettricista inserisce in un quadro e poi collega tra loro mediante dei fili per ottenere una certa funzione complessiva. Questi pezzi sono realizzati mediante la scrittura di un programma che elabora dei segnali di ingresso, definisce dei segnali di uscita e fa uso di dati memorizzati al suo interno.
Il blocco funzione necessita per la sua esistenza di una **struttura dati** che genericamente raccoglie i valori di **ingresso**, i valori di **uscita** ed i valori **interni** al blocco. Questa struttura dati non è altro che una porzione della memoria ram del PLC (anche con permanenza allo spegnimento) che viene dedicata appositamente ad ogni utilizzo di un blocco funzione o meglio ad ogni **istanza** di applicazione dell'oggetto. Quindi ogni volta che si inserisce nel quadro uno di questi componenti, si riserva una specifica e distinta area della memoria ram del PLC per appoggiare tutto quanto

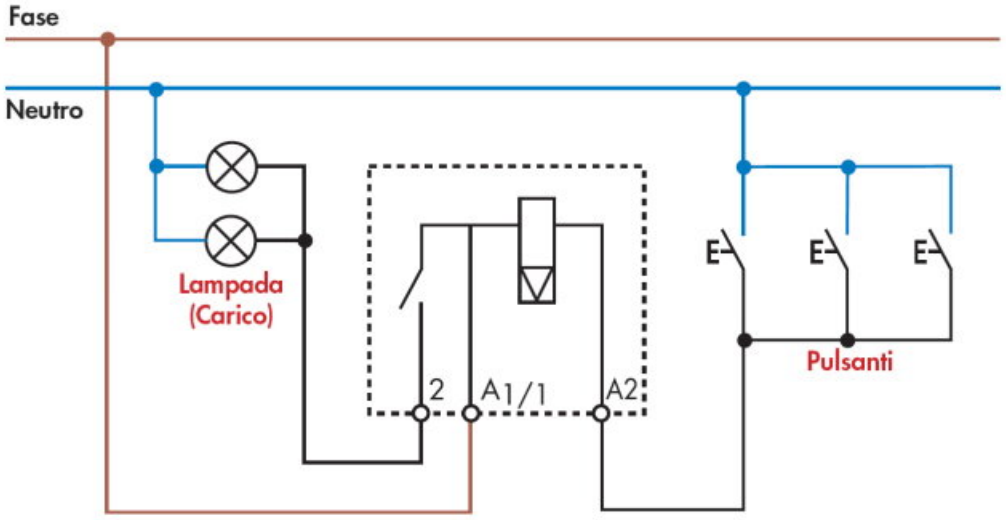
serve al lavoro del blocco funzione. Dobbiamo quindi immaginare come se il blocco funzione fosse una **scatola chiusa** con all'interno quest'area di memoria sua propria. In una parte dell'area viene copiato il valore corrente dei suoi morsetti di ingresso, in una parte sono definiti i valori delle uscite da prelevare ai morsetti e nel resto dell'area ci sono diversi valori intermedi e di calcolo necessari alla funzione. Infine dentro lo scatolotto c'è pure un programma che elabora tutti questi valori secondo una precisa logica.
Ad essere precisi il listato di tale programma ha il vantaggio di essere definito e caricato nel PLC una sola volta per tutte le istanze di applicazione del blocco. Semplicemente passando il riferimento all'area si fa lavorare lo stesso codice programma su tutte le istanze del blocco. Quindi alla fine ogni oggetto usato **consuma risorse** solo per la propria area ram di memoria dati. Se si pensa a questa descrizione del blocco funzione e a come funziona un PLC, viene quasi da dire che il blocco funzione è come un **piccolo PLC**, con una suo programma ben preciso, inserito dentro un **PLC complessivo**. Non a caso la programmazione standard IEC è strutturata in modo da scomporre un grande impianto d'automazione in tanti più piccoli impianti connessi tra loro.
La definizione, una volta per tutte, di un blocco funzione crea un oggetto utile e versatile che può essere riciclato più volte nella stessa applicazione o anche in altre completamente diverse. Un comportamento ben definito, autosufficiente e ben documentato del blocco funzione sono elementi importanti per la **riciclabilità dei componenti virtuali**. La creazione di questi oggetti funzionali prescinde quindi dall'applicazione specifica e per questo hanno un elevato valore in termini di sviluppo e progettazione, proprio per la loro riusabilità nonchè rivendibilità. Creare oggetti virtuali con una bella funzione ed utilizzabili su diversi PLC e su diverse applicazioni può anche diventare interessante.

Il relè passo-passo

Il relè passo-passo è un componente molto utilizzato nell'impiantistica civile. Ideato e **brevettato nel 1949** da Piero Giordanino, poi fondatore di Finder, è un particolare dispositivo che può comandare ciclicamente l'accensione e lo spegnimento di una lampada con la semplice pressione di un pulsante. Il meccanismo si basa su una **camma** a sezione quadrata che viene fatta ruotare dall'attivazione, mediante il pulsante, della bobina di un elettromagnete. Questa camma aziona la posizione del contatto di uscita ciclicamente tra la posizione di aperto e di chiuso. Il relè passo-passo si presenta come una piccola scatola dotata di pochi morsetti:

Relè passo-passo elettromeccanico

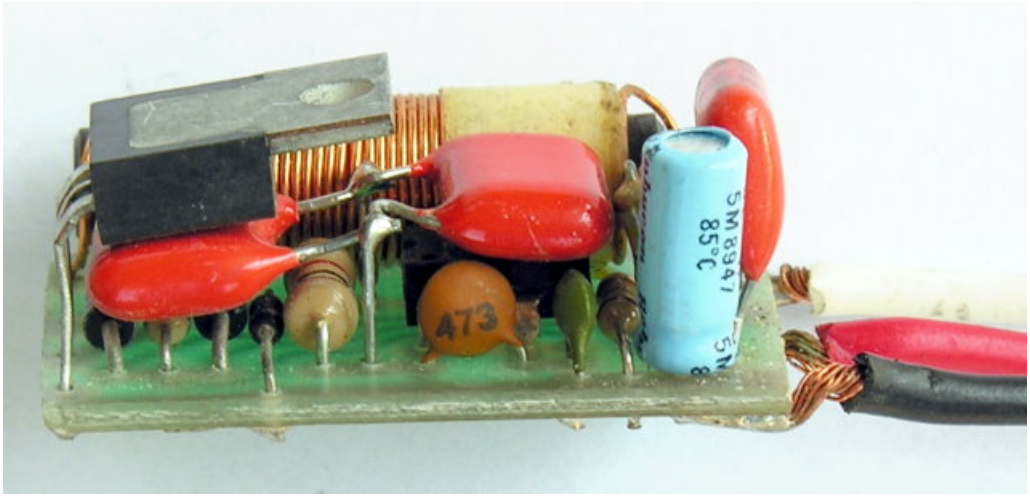
Il principale vantaggio derivante dall'utilizzo del relè passo-passo consiste nella possibilità di comandare l'accensione e lo spegnimento della lampada **da più punti della stanza** semplicemente collegando altrettanti pulsanti di comando in parallelo, evitando così i complessi cablaggi di deviatori ed invertitori. Inoltre l'attivazione della bobina del relè avviene con una **corrente ben più bassa** di quella del carico e quindi la sezione dei cavi, ai vari punti di comando, può essere ridotta.
Un tipico schema di un impianto con relè passo-passo è il seguente:



Connessione del relè passo-passo elettromeccanico

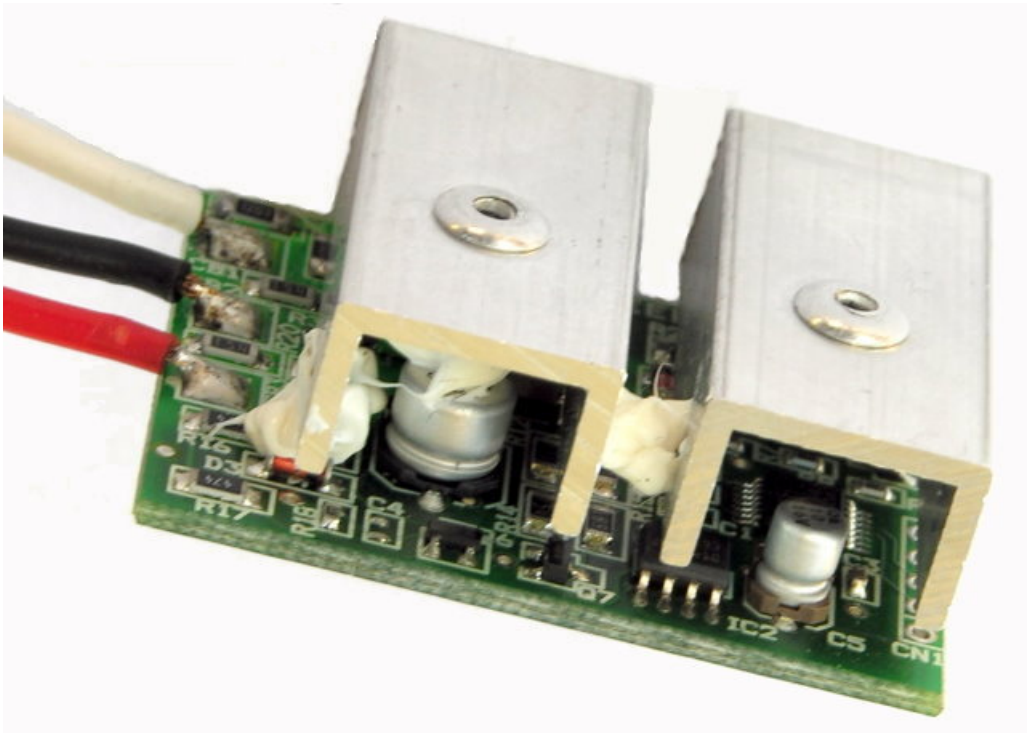
Il grande successo di tale componente è sempre stato inseguito da tanti produttori di materiale per impiantistica civile, proponendo soluzioni alternative anche ricorrendo all'**elettronica**. Ovviamente questo è stato possibile solo in tempi più recenti, quando, per esempio, sono stati realizzati **circuiti integrati** dedicati a tale funzione. Uno dei più vecchi integrati era stato introdotto da Siemens, l'SLB0586, poi seguito dal LS7232 ed altri. Questo componente esegue la funzione di ON/OFF della lampada utilizzando un **interruttore statico** (un triac) al posto del contatto meccanico.

Inoltre, ormai che c'erano, hanno inserito pure la funzione di variazione della luminosità, sfruttando la regolazione a **parzializzazione di fase**. La seguente foto mostra un relè passo-passo d'epoca utilizzante tale integrato:



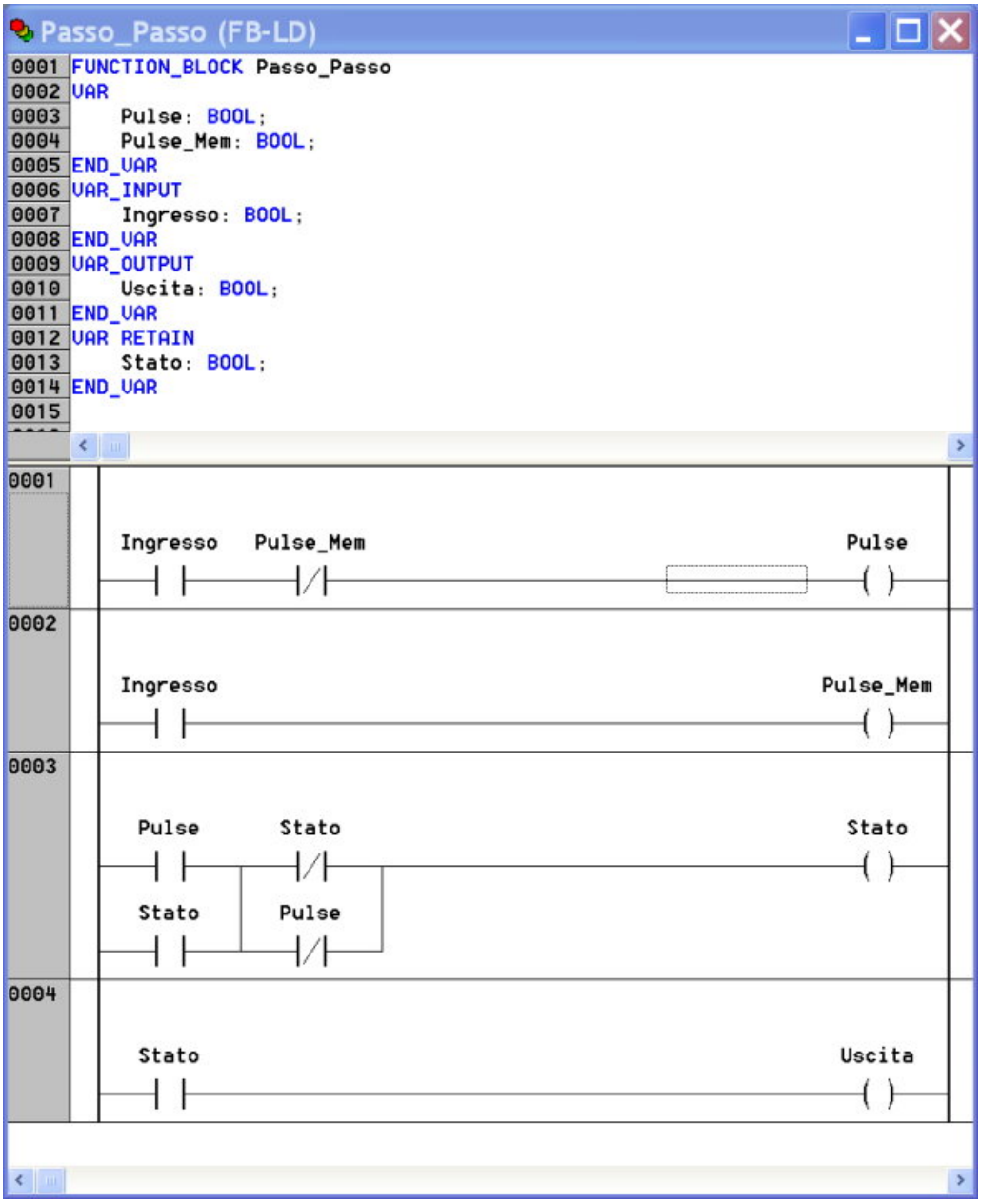
Relè passo-passo elettronico antico

Questa foto invece mostra un circuito più recente a **microprocessore**, con vari programmi e funzioni, nonchè il pilotaggio del carico AC mediante un **doppio mosfet**:



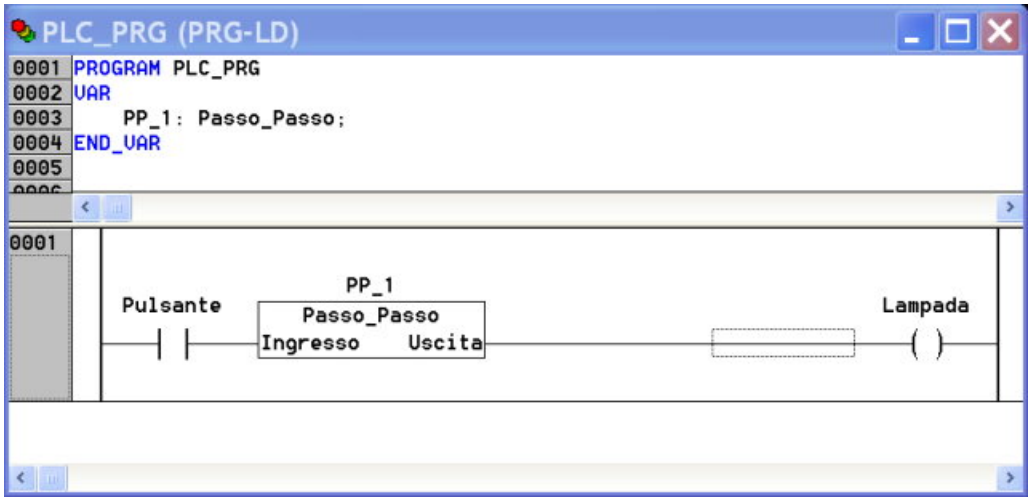
Relè passo-passo elettronico a microprocessore e mosfet

L'introduzione di sistemi programmabili per la domotica non poteva ovviamente ignorare questo usatissimo componente. Per questo ho scelto proprio il relè passo-passo come primo esempio di creazione di un blocco funzione. Disponendo di un PLC con solo uscite digitali mi sono limitato ad implementare la sola classica funzione di ON/OFF, rimandando la regolazione della luminosità ad un più versatile blocco funzione che utilizzerà il **protocollo DMX512** sull'interfaccia seriale del PLC.
Finalmente, dopo tanto divagare, arrivo al dunque. Ecco tutto il listato programma in linguaggio ladder (LD) del blocco funzione del passo-passo:



Listato ladder del blocco funzione passo-passo

Un qualsiasi modulo programma è sempre diviso in due parti. In alto tutte le **dichiarazioni delle variabili locali** del modulo ed in basso il vero e proprio **listato programma**. Le dichiarazioni possono essere elencate ed editate sia nella forma di testo, come nell'esempio, sia in forma di tabella, divisa in righe per ogni variabile e in colonne per le relative specifiche. Le variabili sono poi suddivise in gruppi: variabili generiche locali, variabili di ingresso, di uscita e di tipo Retain, ossia a mantenimento del valore a PLC spento.
Questo listato programma è fatto solo da 4 rami elettromeccanici. I primi due formano il classico **generatore di un impulso** ogni volta che la variabile **Ingresso** ha un fronte di salita (passaggio da OFF ad ON). Infatti il bit di appoggio **Pulse** diventa ON solo se il corrente valore di Ingresso è ON mentre il valore precedente (o meglio al ciclo di scansione precedente) era OFF. La seconda rete serve appunto per salvare nel bit di appoggio Pulse_Mem questo valore di Ingresso per il ciclo successivo. Questi primi due rami potevano anche essere rimpiazzati dal blocco funzione standard già disponibile per rilevare il fronte di salita, ma ho voluto fare tutto con pezzi elementari.
Il ramo 3 è quello che gestisce la **funzione del passo-passo**. Per lo stato corrente del relè (acceso/spento) ho utilizzato il bit di appoggio **Stato** di tipo Retain, in quanto volevo mantenere lo stato selezionato di ON/OFF anche quando viene a mancare l'alimentazione elettrica (per esempio per un blackout). Quindi il compito del circuito elettromeccanico è quello di commutare il valore ON/OFF del bit Stato ogni volta che la pressione del pulsante sull'ingresso causa la generazione di un impulso sul bit Pulse. Infatti, nella parte alta del ramo 3, abbiamo la serie del segnale Pulse con il negato del segnale Stato. Quindi il bit di uscita Stato va ON solo se esso era OFF ed arriva un impulso di pressione del pulsante. Quando poi il bit Stato è diventato ON, ma l'impulso è terminato, avviene, con la parte sottostante del ramo 3, un'autoritenuta del valore ad ON di Stato. Alla prossima pressione del pulsante il contatto negato di Pulse, andando OFF, farà cadere questa autoritenuta, riportando il bit Stato ad OFF. Infine il bit a memoria permanente Stato è copiato sul bit **Uscita** del blocco funzione mediante il ramo 4.
Ora non rimane che **utilizzare questo blocco funzione** in un programma applicativo di prova. Per prima cosa dichiariamo la necessità di far uso di tale componente. La dichiarazione di un'istanza di utilizzo del blocco equivale alla dichiarazione della struttura dati ram associata di cui parlavo in precedenza. Poi lo utilizziamo collegando all'ingresso il contatto **Pulsante** (nome simbolico che ho associato, nel PLC configuration, all'ingresso In_0) e collegando all'uscita la bobina del relè che accende la lampada (ho nominato col simbolo **Lampada** il bit di uscita fisica Out_0). Il listato applicativo, anch'esso in linguaggio ladder (ma il blocco funzione può essere utilizzato anche in programmi scritti con gli altri linguaggi) è il seguente:



Utilizzo del componente relè passo-passo

Per finire un brevissimo video in HD su uno **speciale relè passo-passo**, studiato per l'illuminazione di sale adatte a cene d'occasione:

Flash

Il luce-scale con avviso di spegnimento

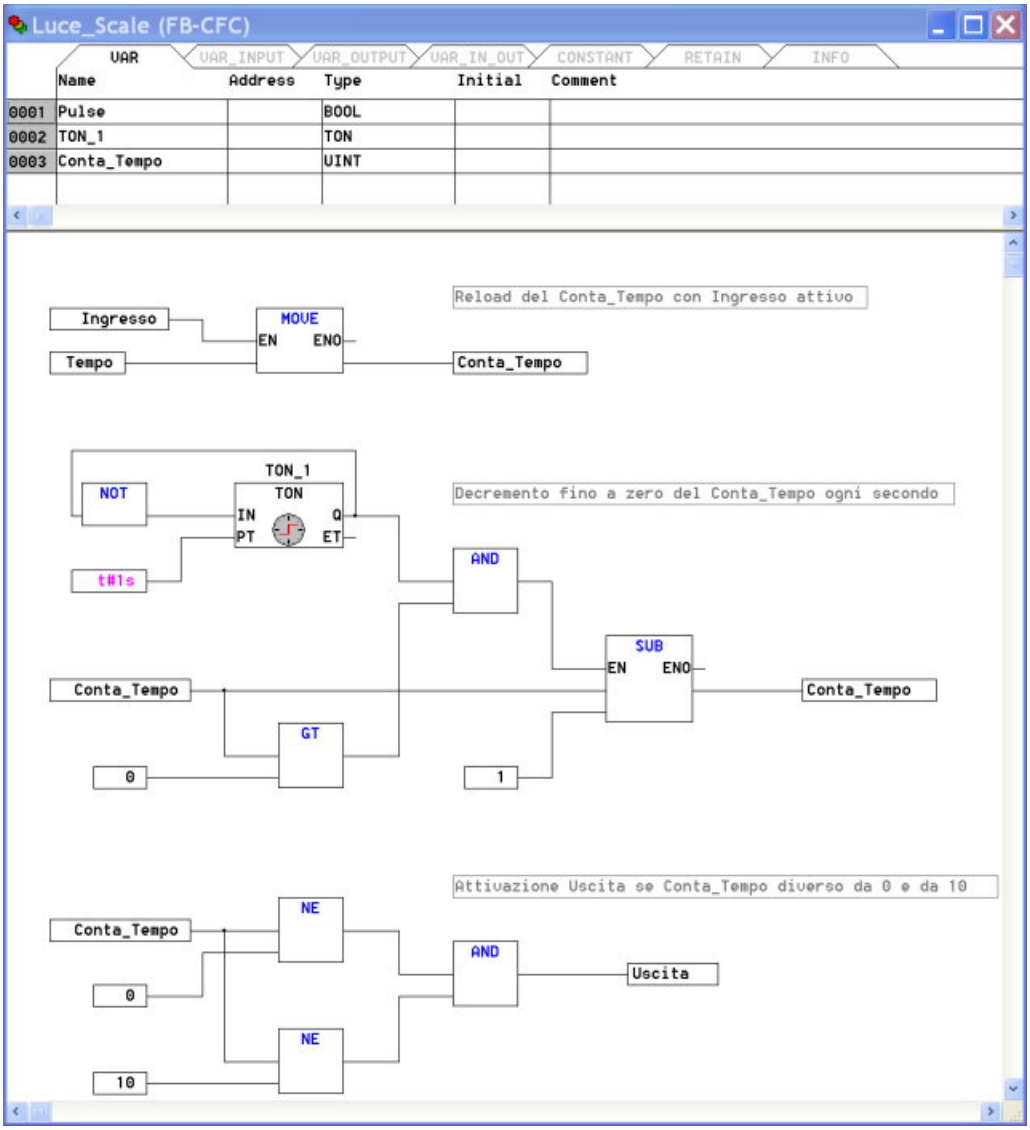
Il luce-scale è un'altro apparecchio che da tempo viene impiegato all'interno dei quadri elettrici ad uso domestico. Il suo utilizzo è molto diffuso in costruzioni a più piani dotate di **lunghe rampe di scale** come i **palazzi condominiali**. Questo dispositivo è praticamente un **temporizzatore** attivabile mediante più pulsanti posti in uno o più punti di ogni piano, con i quali è possibile accendere l'illuminazione delle

rampe di scale. La funzione del luce-scale in parte richiama quella del relè passo-passo, in quanto permette l'accensione delle lampade da molti punti con la semplice connessione di **tanti pulsanti in parallelo**, semplificando l'impianto elettrico. Tuttavia il luce-scale è caratterizzato dalla funzione di **spegnimento automatico** dopo un certo tempo impostato. Questa funzione assume una grande importanza per quanto riguarda i consumi di elettricità, riducendo di molto i costi delle bollette condominiali.
Lo spegnimento automatico delle lampade però crea un certa difficoltà all'utente quando, improvvisamente, terminato il tempo, rimane al buio. Per questo sono state introdotte varie tecniche di segnalazione del **tempo quasi al termine**. Per esempio, apparecchi più elettronici effettuano una dissolvenza graduale dell'illuminazione allo scadere del tempo, mentre altri più semplicemente avvertono l'utente, mediante un **brevissimo buco di luce**, che si deve ripremere il pulsante per riattivare dall'inizio la temporizzazione. Nell'esempio successivo di implementazione del componente luce-scale, mediante un blocco funzione, ho adottato questo tipo di segnalazione.
Il luce-scale può comunque essere utilizzato anche per temporizzare l'illuminazione, non solo di rampe di scale, ma anche di **garages** e **giardini**, magari comandandoli, oltre che con dei pulsanti, anche con dei sensori di presenza. Degli esempi di apparecchi luce-scale sono i seguenti:

Alcune immagini di apparecchi luce-scale elettronici

Per implementare il blocco funzione del luce-scale utilizzerò, tanto per fare un esempio, uno degli altri linguaggi di tipo grafico permessi dal tool di programmazione. Ricordo che lo standard IEC61131-3 prevede un totale di **5 diversi linguaggi** dei quali 2 di tipo testuale (Instruction List **IL** e Structured Text **ST**) e 3 grafici (Ladder **LD**, Function Block Diagram **FBD** e Sequential Function Chart **SFC**). In particolare il linguaggio FBD descrive la funzione di programma mediante uno schema elettronico realizzato dalla connessione di componenti o blocchi funzionali. Lo schema complessivo è realizzato dall'insieme di pezzi di schema un po' come avviene per i rami elettromeccanici dello schema ladder.
Il tool di programmazione CoDeSys implementa un **sesto linguaggio**, equivalente al FBD, ma con la particolarità che lo schema di interconnessione dei componenti possa essere più completo e con connessioni anche di retroazione. Con questo linguaggio, chiamato Continuous Function Chart **CFC**, si rappresenta il programma in modo

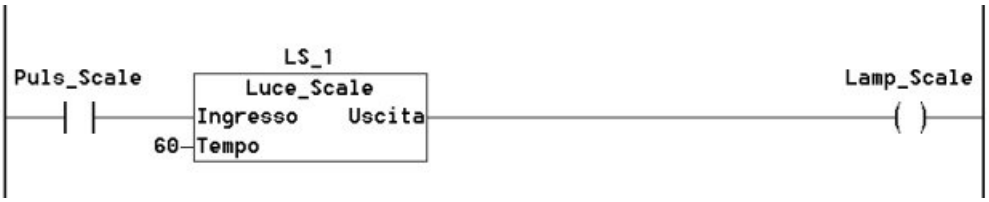
ancora più simile ad un **completo schema elettronico** su uno o più grossi fogli bianchi invece che con pezzi di schema successivi.
Lo schema completo, in linguaggio CFC, del blocco funzione del luce-scale con avviso di spegnimento è il seguente:



Il programma a schema grafico del blocco funzione luce-scale

Lo schema è più semplice di quanto sembri ed ora proverò a spiegarlo. Prima di tutto è stato spezzato in **tre parti**, visivamente sconnesse ma facenti parte di uno stesso schema. La funzione complessiva si basa principalmente sulla gestione di una variabile locale, di tipo intero senza segno (a 16 bits) chiamata **Conta_Tempo**. In questa variabile ci scriviamo un valore, passato come parametro di input al blocco funzione e chiamato **Tempo**, ogni volta che attiviamo il segnale **Ingresso** del blocco (connesso, nel suo utilizzo ai pulsanti di accensione del luce-scale). Quindi fino a

quando premiamo uno dei pulsanti il valore Conta_Tempo è continuamente forzato al valore del tempo impostato (per esempio 60 per 60" di temporizzazione). Questo precaricamento del Conta_Tempo avviene mediante la prima parte dello schema con l'attivazione (segnale di enable EN) del **blocco funzione standard MOVE**.
Nella parte centrale dello schema CFC abbiamo il cuore della **temporizzazione** del luce-scale. Innanzitutto serviva un **segnale di clock** di riferimento tempo che è stato ottenuto con un blocco funzione standard di tipo **timer TON** (timer ritardato all'accensione). Questo timer, quando alimentato al suo ingresso inizia a conteggiare un tempo fisso impostato a **1"** al termine del quale attiva ON la sua uscita. Appena l'uscita va ON il segnale di alimentazione del timer viene a mancare in quanto pilotato dal segnale negato della stessa sua uscita, resettandolo e riavviando il tempo dall'inizio. Così il segnale di clock conterrà un impulso ON breve e ripetuto ogni secondo. La restante parte centrale viene quindi attivata dal segnale di clock ogni secondo per **decrementare di 1** il registro Conta_Tempo, ma solo se questo valore è ancora **maggiore di 0**. Per questo sono utilizzati il blocco funzione standard di comparazione **GT** (greater then) rispetto al valore **costante 0** ed un blocco funzione **SUB** (subtract) del valore **costante 1**. Gli impulsi di clock abiliteranno la funzione di sottrazione solo se (funzione **AND**) la comparazione Conta_Tempo > 0 è verificata.
Infine la terza parte è quella che definisce il valore dell'uscita del blocco funzione, connessa poi alle lampade delle scale. Una volta precaricato il registro Conta_Tempo con il numero di secondi richiesti, al rilascio del pulsante, il valore Conta_Tempo comincia a decrementare di 1 ogni secondo (conto alla rovescia) fino a fermarsi a zero. Allora noi **intercetteremo** solo i due valori 0 e 10 dei secondi rimasti per decidere di spegnere la luce. Così avremo un **buco di luce a 10"** dal termine del tempo, della durata di 1", e ovviamente la luce definitivamente **spenta al termine del tempo**. Quindi con un paio di funzioni standard di comparazione **NE** (not equal) del valore Conta_Tempo con le costanti 0 e 10, otteniamo due distinti segnali attivi ON se Conta_Tempo è diverso da 0 e da 10. Solo se entrambe le condizioni sono verificate, il blocco funzione **AND** darà un segnale ON per attivare l'**Uscita** del blocco funzione luce-scale.
Una volta definita la funzione interna al blocco funzione luce-scale, possiamo dimenticarcela ed utilizzare il blocco come un componente finito, con una funzione ben precisa scritta sul suo foglietto di istruzioni allegato:

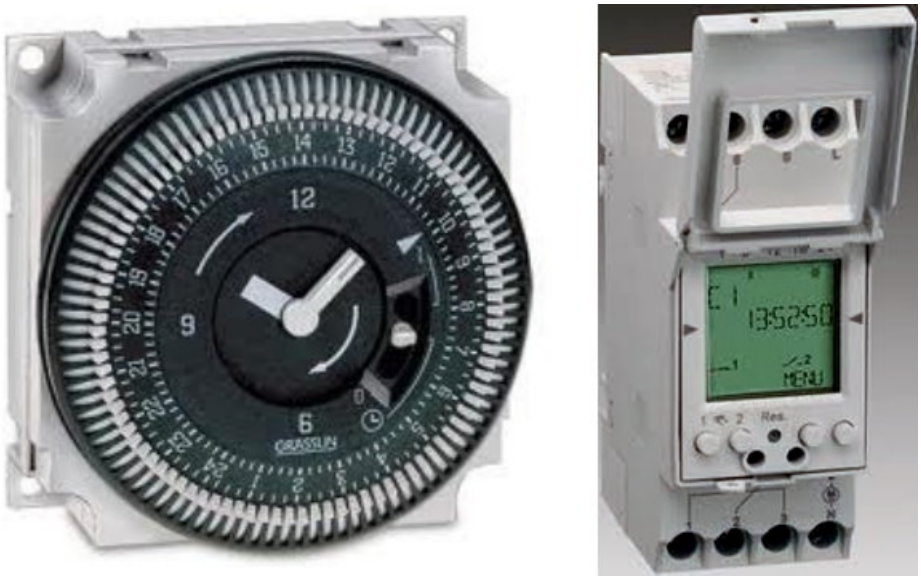


Utilizzo del blocco funzione luce-scale

In questo esempio applicativo è stato nuovamente utilizzato il linguaggio grafico ladder che molto bene rappresenta lo schema elettrico di collegamento del luce-scale. Infatti l'ingresso In_1 del PLC (il successivo rimasto libero dopo l'esempio del passo-passo), ribattezzato col nome simbolico **Puls_Scale** attiva l'ingresso del luce-scale. L'uscita del dispositivo è poi connessa al relè di uscita Out_1 ribattezzato **Lamp_Scale**. Infine il blocco prevede come parametro di input il **Tempo** voluto di autospegnimento della luce, collegato in questo esempio alla costante 60 (secondi). Il valore di tempo tuttavia, invece che essere costante, può provenire da una variabile impostabile per esempio mediante l'interfaccia operatore con Web-server. In questo modo si può modificare il valore del tempo accedendo con un browser Internet alla **pagina Web** del quadro elettrico.

Un programmatore orario

Il programmatore orario è un oggetto molto conosciuto anche dall'utilizzatore finale dell'impianto elettrico di casa (ossia da chi ci abita). Chi non ha mai avuto a che fare con i **chiodini** o **levette** dei programmatori elettromeccanici, o chi non si è mai trovato ad armeggiare con i **pulsanti** e gli **enigmatici menu** di un programmatore orario elettronico con display lcd? Questi oggetti o componenti sono presenti sia in versione da quadro elettrico che in versione "spiccia", come un piccolo elettrodomestico. Alcuni esempi di questi componenti sono riportati nella seguente figura:



Un programmatore orario elettromeccanico ed uno elettronico

Ora implementeremo mediante un blocco funzione anche questo componente, così potremo contare sulla disponibilità di un elevato numero di questi apparecchi, pagandoli sempre la stessa cifra, alla faccia del prezzo non sempre basso di tali

componenti. Ovviamente, per semplicità della mia spiegazione, mi limiterò ad un programmatore orario **molto semplice** (giornaliero e non settimanale e con sole due fasce orarie). Tuttavia, una volta capito il meccanismo della creazione del blocco funzione, l'ampliamento al **settimanale e più fasce orarie** è praticamente immediato. Mi preme infatti, a favore della chiarezza, evitare di fare confusione con la ripetizione a raffica, per tutti i giorni della settimana, dello stesso identico pezzo di listato programma.
Per questo esempio non userò un linguaggio grafico, come ho fatto in precedenza, ma un più classico linguaggio di programmazione a testo: lo **Structured Text** abbreviato con **ST**. Questo linguaggio è spesso paragonato al **Pascal**, un linguaggio molto istruttivo (usato ai miei tempi all'università e che ricordo ben volentieri), anche se a mio avviso, l'assomiglianza con il **Basic** è notevole. Comunque non ci interessa di chi è figlio, basta che sia potente e flessibile e su questo non ci piove, visto che è molto utilizzato dai programmatori che provengono dal di fuori del mondo dei PLC ed abituati ai noti linguaggi di programmazione.
Come regola, prima di iniziare a scrivere le istruzioni interne al blocco, occorre aver ben chiara la funzione che dovrà svolgere, quasi dovessimo scrivere il **foglietto di istruzioni** all'uso ancor prima di aver realizzato il componente. Quindi occorre anche definire quelli che sono gli input/output dell'apparecchio, ossia i morsetti di connessione. Il programmatore orario dovrà avere due distinte ed indipendenti fasce orarie all'interno delle quali si dovrà accendere l'uscita. Inoltre lo doteremo di un **ingresso di manuale** per attivare direttamente l'uscita anche fuori dagli orari impostati. Dunque il listato contenuto, direi chiuso dentro la scatola avvitata del programmatore e del quale non dovremo mai più impicciarci (altrimenti non sarebbe un componente finito) è il seguente:

Prog_Orario (FB-ST)

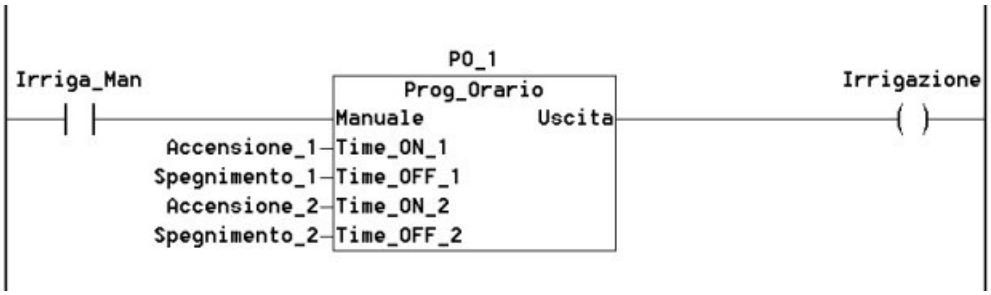
```
0001 FUNCTION_BLOCK Prog_Orario
0002 VAR
0003     Current_Date_Time: DATE_AND_TIME;
0004     Current_Time:      TIME_OF_DAY;
0005 END_VAR
0006 VAR_INPUT
0007     Manuale:    BOOL;
0008     Time_ON_1:  TIME_OF_DAY;
0009     Time_OFF_1: TIME_OF_DAY;
0010     Time_ON_2:  TIME_OF_DAY;
0011     Time_OFF_2: TIME_OF_DAY;
0012 END_VAR
0013 VAR_OUTPUT
0014     Uscita: BOOL;
0015 END_VAR
0016
0017
```

```
0001 (* Lettura dell'ora corrente dall'orologio di sistema *)
0002
0003 Current_Date_Time := DateTime_Read();
0004 Current_Time      := DT_TO_TOD(Current_Date_Time);
0005
0006
0007 (* Definizione dell'uscita in base alle due fasce orarie *)
0008
0009 IF Manuale THEN
0010     Uscita := TRUE;
0011 ELSIF (Current_Time >= Time_ON_1) AND (Current_Time <= Time_OFF_1) THEN
0012     Uscita := TRUE;
0013 ELSIF (Current_Time >= Time_ON_2) AND (Current_Time <= Time_OFF_2) THEN
0014     Uscita := TRUE;
0015 ELSE
0016     Uscita := FALSE;
0017 END_IF
0018
0019
```

Listato Structured Text del blocco funzione prog-orario

Per prima cosa è necessario disporre di un orologio per sapere l'ora corrente esatta, altrimenti è inutile andare avanti. A questo pensa la funzione **DateTime_Read** che restituisce la data e l'ora corrente nella variabile **Current_Date_Time**. Esistono vari modi standard per richiedere la data e l'ora al PLC, ma ho voluto predisporre questa funzione IEC, tutta mia e molto diretta, creandola in C e poi "bruciandola" dentro sistema operativo della EY-CPU. La variabile Current_Date_Time in questione è dichiarata del tipo **DATE_AND_TIME** facente parte dello standard IEC. Una variabile di questo tipo permette di contenere l'informazione della data e dell'ora mediante con il solito trucco del numero di secondi trascorsi da una data di riferimento (tipo primo gennaio del 1970). Chi volesse dare un'occhiata al sorgente in C della funzione e ragionare con gli **anni bisestili** può scaricare questo **RTC_lib.c** che può essere utile in generale. Tuttavia a noi non interessano i secondi o i meccanismi nascosti dal sistema, ma ci interessa sapere l'esatta ora del giorno. Per questo ho utilizzato una funzione **DT_TO_TOD**, standard IEC, per convertire il tipo

data/ora nel tipo **TIME_OF_DAY**. Quindi la variabile **Current_Time** ora conterrà la sola informazione dell'ora corrente del giorno.
Tra le variabili di ingresso del blocco ho dichiarato il bit **Manuale**, per bypassare manualmente il programmatore orario, e le preimpostazioni degli orari di inizio e fine delle due fasce orarie: **Time_ON_1** e **Time_OFF_1** per la prima fascia e **Time_ON_2** e **Time_OFF_2** per la seconda. A queste quattro variabili, dichiarate di tipo TIME_OF_DAY, saranno passati i valori di altrettante variabili del programma applicativo, preimpostate all'accensione con valori costanti tanto per definirne un default. Sarà poi compito del software applicativo di renderle modificabili a piacere, per esempio associandole ad altrettante caselle di una pagina grafica del Web-server, così collegandoci col browser del PC possiamo modificarle a piacere.
La spiegazione del listato programma è quasi banale. Ho utilizzato la classica istruzione **IF/ELSEIF** per confrontare l'ora corrente con gli orari di accensione e spegnimento impostati. Solo se l'ingresso Manuale è ON, altrimenti se siamo all'interno della prima fascia oraria, altrimenti se siamo all'interno della seconda fascia oraria, allora l'uscita è attiva ON, altrimenti è sempre OFF. Notare la comparazione diretta tra due orari senza bisogno di "impelagarsi" con le ore ed i minuti. Il listato del programma applicativo del blocco funzione è altrettanto banale.
Come al solito ho utilizzato il linguaggio ladder per l'esempio applicativo del blocco:



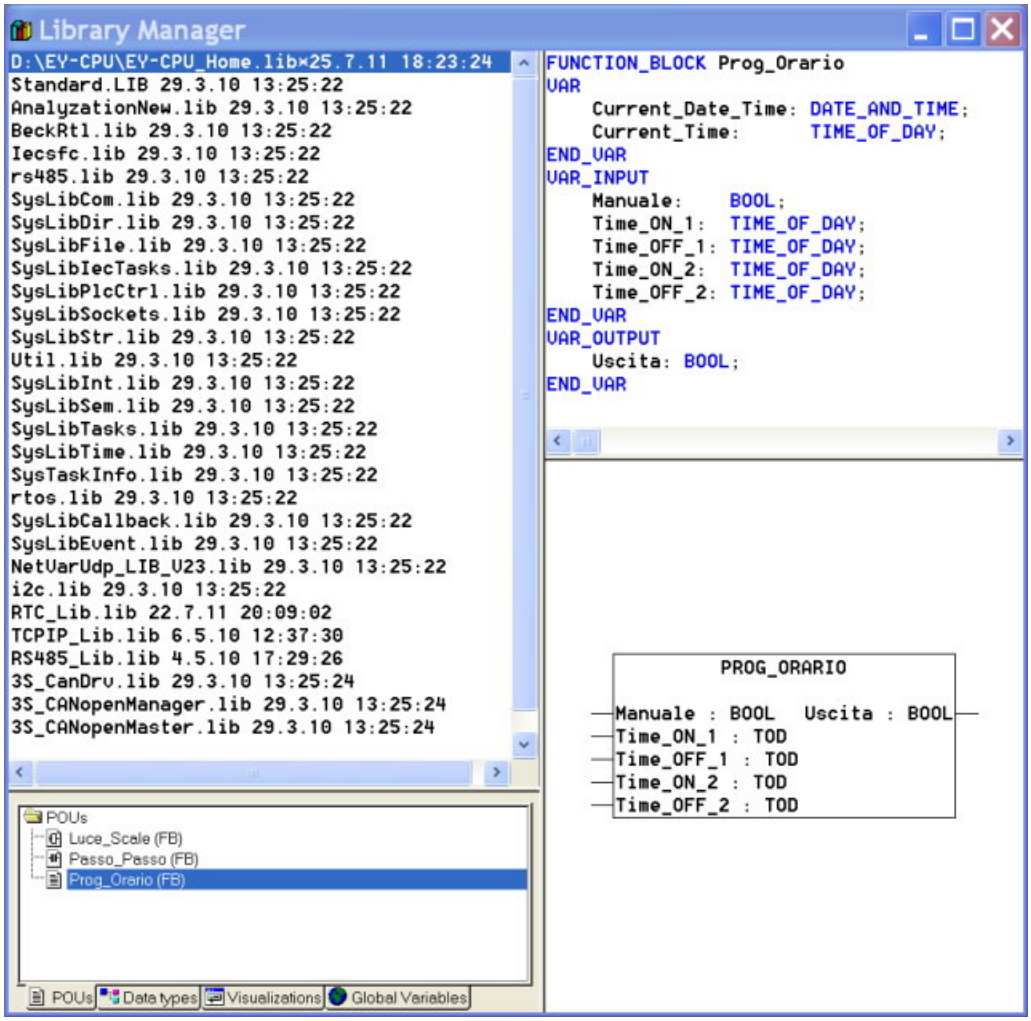
Utilizzo del blocco funzione prog-orario

Ho definito il simbolo **Irriga_Man** per l'ingresso digitale su morsettiera In_2, passandolo come ingresso al blocco. Inoltre ho dichiarato **quattro variabili** di tipo TIME_OF_DAY nella memoria Retain, così non le perdo una volta editate col Web-server, passandole anch'esse come ingresso all'istanza di utilizzo del blocco funzione. Infine ho associato al relè di uscita Out_2 il simbolo **Irrigazione** e lo ho connesso all'uscita del blocco. Quindi il relè di uscita del PLC si accenderà tutti i giorni all'interno delle due fasce orarie programmate.

Raccogliamo gli oggetti nella libreria domotica

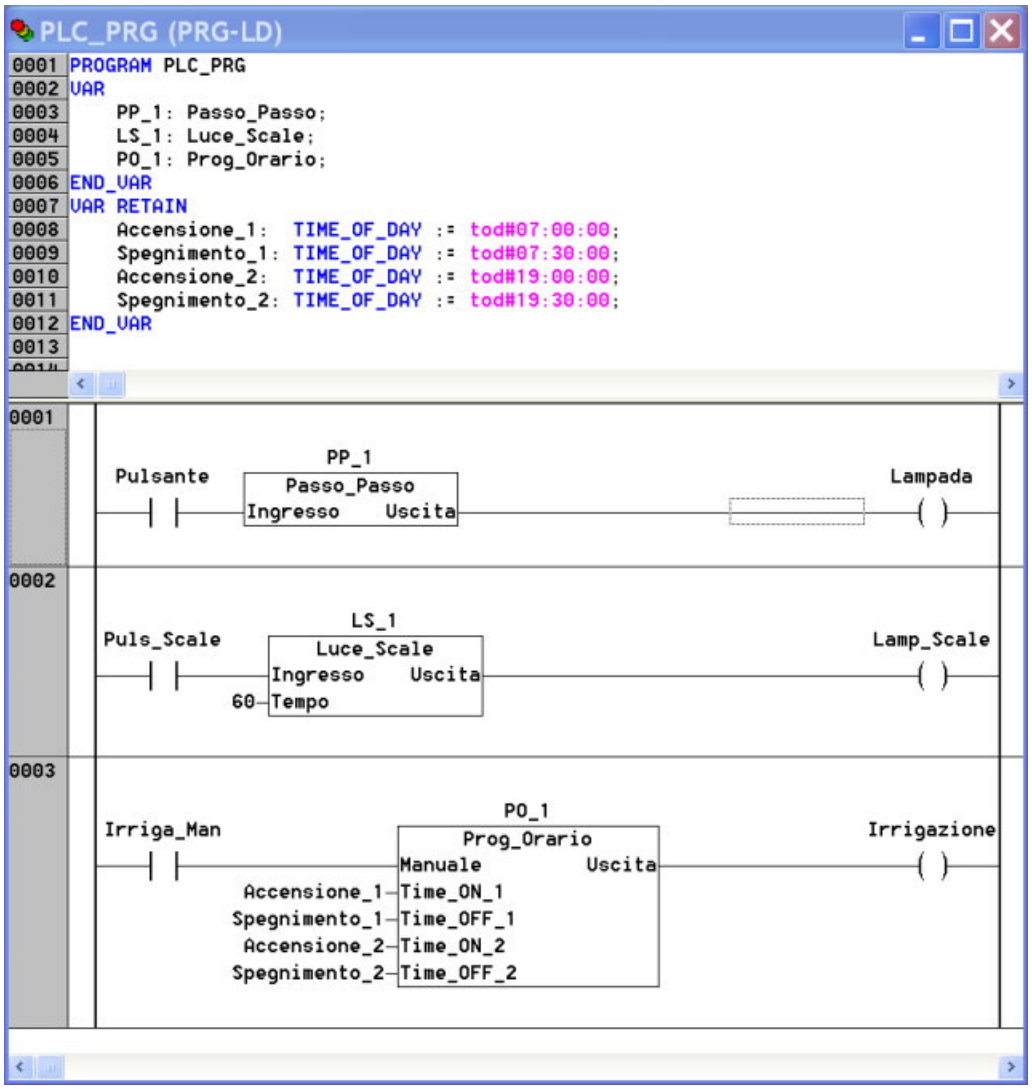
Come ho più volte detto la potenza del linguaggio IEC e dei blocchi funzione è determinata dalla **riciclabilità del software**. Per questo, una volta fatto lo sforzo di creare dei blocchi funzione, voglio collezionarli definitivamente in una **libreria**,

in modo da poterli riutilizzare come componenti finiti in altre applicazioni. In questo modo il lavoro fatto assume una grande importanza poichè ci risparmierà del lavoro per il seguito e ci eviterà di ripercorrere ogni volta sempre le stesse problematiche.
Per creare una nuova libreria basta partire da un qualsiasi progetto, lasciando solo le tre POU relative alla definizione dei blocchi funzione creati e salvando il progetto (Save As) come file **.lib**, appunto di libreria. In un futuro progetto poi basterà caricare con il **Library Manager** la nostra libreria:



La libreria domotica con i tre componenti creati

Come si vede la libreria è stata aggiunta ad un insieme di altre librerie, standard o specifiche del PLC, ciascuna che mette a disposizione un certo numero di funzioni e blocchi funzione già pronti all'uso. Ora, tutto ciò che ho raccontato in tale articolo ed i relativi listati riportati, potrebbero anche essere dimenticati. I tre oggetti creati sono come dei componenti finiti la cui funzione è nota e dei quali non ci interessa più sapere come sono fatti dentro. Quindi possiamo scrivere finalmente il nostro **primo programma applicativo** che si riduce a solo questo breve listato ladder:



Il programma applicativo completo

Ovviamente, come vanno dichiarate le variabili che si intende utilizzare nel programma, **vanno dichiarate anche le istanze** di applicazione dei blocchi funzione. D'altra parte, come dicevo in precedenza, le risorse hardware necessarie all'esistenza dei blocchi funzione, non sono altro che delle variabili, anche di diverso tipo, raccolte in un'unica struttura dati che va dichiarata come una qualsiasi semplice variabile.
Ecco i files del programma CoDeSys e della libreria creata:

Download

EY-CPU_Home.zip

Conclusione

In questo articolo ho voluto utilizzare un argomento molto attuale, appunto la domotica, come spunto pratico per parlare di programmazione IEC ed illustrare almeno **3 dei 6 linguaggi** previsti dall'ambiente di programmazione. Per questo i miei **3 blocchi funzione** non devono essere assolutamente considerati come lo stato dell'arte della domotica, ma come dei semplici e banali esempi di programmazione. Ho voluto inoltre dedicare una particolare attenzione alla scomposizione di un problema più complesso in parti o moduli ricorrenti fatti coincidere con i blocchi funzione.
Anche se poi alla fine passo comunque tante ore lo stesso davanti al computer, la **mia pigrizia** mi ha sempre imposto di ricercare e mettere a punto i mattoni elementari sui quali basare le mie prossime costruzioni, evitando il più possibile di ripercorrere sempre le stesse fatiche. Nel fare un progetto hardware o software **mai pensare solo al progetto stesso** ma a quali sono le sue parti costitutive e le relative soluzioni che prescindono da questo, sognando ad occhi aperti di vederle già applicate in ipotetici e successivi progetti, quasi dimenticandosi di quello presente. Spero che questo approccio sia stato abbastanza evidente in tutte le fasi del mio progetto ad iniziare dalla realizzazione dell'hardware fino ad arrivare a questo articolo. La scheda realizzata potrebbe avere già di per se stessa diversi utilizzi, ma per me la cosa più importante è aver avuto l'occasione di mettere insieme e testare parti hardware e software, nonché idee e migliorie, che entreranno a far parte di quel piccolo bagaglio a mano che si deve portare sempre appresso un progettista.
Ora il mio piccolo PLC vola in alto quasi come un vero e grande PLC, ma è meglio che lo riporti con i piedi sulla terra ricordandogli che è soltanto una **piccola**, **brutta** e **nuda** scheda elettronica...