

# 0. Introduzione al PLC

## 0.1 Controllore Logico Programmabile (PLC)

Il **controllore logico programmabile** o *Programmable Logic Controller (PLC)* è un computer utilizzato nella gestione dei processi industriali. Il PLC esegue un programma elaborando i segnali digitali e analogici provenienti da sensori e diretti agli attuatori presenti in un impianto industriale. Grazie alla diminuzione dei costi, è entrato anche nell'uso domestico. L'installazione di un PLC nel quadro elettrico di un'abitazione permette di gestire automaticamente sistemi come riscaldamento, antifurto, irrigazione, LAN, luci, ecc.

Diamo adesso la definizione secondo lo standard europeo:

**PLC (Programmable Logic Controller)** – Sistema elettronico a funzionamento digitale, destinato all'uso in ambito industriale, che utilizza una memoria programmabile per l'archiviazione interna di istruzioni orientate all'utilizzazione per l'implementazione di funzioni specifiche, come quelle logiche, di sequenziamento, di temporizzazione, di conteggio e di calcolo aritmetico, e per controllare, mediante ingressi ed uscite sia digitali che analogici, vari tipi di macchine e processi.

Un PLC è un oggetto hardware componibile, la cui caratteristica principale è la **robustezza**. Infatti, normalmente il PLC è posto in quadri elettrici soggetti a interferenze elettriche, temperature elevate o umidità. Il PLC è in funzione 24 ore su 24, per 365 giorni all'anno, su impianti che non possono essere arrestati.

## 0.2 Uso del PLC

Il PLC trova applicazione in tutti i contesti dove sono richiesti più di 10 I/O, dove si deve garantire un prodotto affidabile e dove sono previste espansioni o modifiche nella logica di controllo delle apparecchiature industriali, come macchine per stampaggio, imballaggio, tessili, sistemi di movimentazione e trasporto.

La struttura del PLC viene adattata in base al processo da automatizzare. Durante la progettazione del sistema di controllo vengono scelte le schede adatte alle grandezze elettriche in gioco, poi inserite sul bus o rack del PLC.

I PLC hanno sostituito i sistemi a logica cablata, poiché offrono vantaggi come maggiore automazione, funzionalità avanzate e prestazioni simili a quelle dei moderni sistemi a

microprocessore.

## 0.3 Vantaggi dell'uso del PLC: logica programmata

---

La logica cablata collega fisicamente componenti (pulsanti, sensori, relè, contattori) ed è difficile da modificare. La logica programmata presenta invece i seguenti vantaggi:

- **Espandibilità:** moduli componibili che consentono di ampliare la configurazione base.
- **Affidabilità:** sistema statico realizzato con componenti integrati.
- **Flessibilità:** modifiche al processo senza rimuovere il cablaggio, solo aggiornando il programma.
- **Manutenibilità:** diagnostica e ricerca guasti semplificate, con visualizzazione su terminale o stampa.
- **Economicità:** riduce tempi di progettazione, messa a punto e cablaggio, soprattutto per processi complessi.

Il funzionamento logico può essere provato in fase di programmazione tramite simulazione, senza collegare il PLC all'impianto, permettendo diagnostica avanzata e riutilizzo del PLC in altri impianti.

### 0.3.1 Il ciclo di scansione nella logica programmata

I PLC eseguono i programmi in **modo sequenziale**, cioè “*una cosa alla volta, una dopo l'altra*”. Questo principio è alla base del loro funzionamento e influisce sia sul comportamento logico sia sui tempi di elaborazione.

---

#### 0.3.1.2 Concetti chiave del ciclo di scansione

Il **ciclo di scansione** descrive come la CPU esegue le istruzioni e gestisce ingressi, uscite e comunicazioni:

- All'inizio del ciclo, tutti gli ingressi vengono letti simultaneamente e memorizzati nell'**immagine di processo degli ingressi (PII)**.
- Questa immagine rimane **costante per tutto il ciclo**, anche se gli ingressi fisici cambiano stato durante l'esecuzione.
- Le uscite vengono aggiornate alla fine del ciclo, partendo dai valori presenti nell'**immagine di processo delle uscite (PIQ)**.
- La CPU esegue il programma utente, aggiorna moduli di comunicazione e gestisce eventi/allarmi.

 Le richieste di comunicazione e gli allarmi vengono elaborati periodicamente. Eventi di priorità alta possono interrompere temporaneamente l'esecuzione del programma.

---

### 0.3.1.3 Passi del ciclo di scansione

1. **Aggiornamento uscite fisiche:** i valori della PIQ vengono scritti nelle uscite configurate per aggiornamento automatico.
2. **Lettura ingressi fisici:** i valori vengono memorizzati nella PII, garantendo coerenza durante l'esecuzione del programma.
3. **Esecuzione logica utente:** le istruzioni aggiornano i valori di uscita nell'immagine PIQ senza alterare immediatamente le uscite fisiche, evitando instabilità.
4. **Elaborazione combinatoria:** la CPU accede alla PII e ad altre aree di memoria (blocchi dati, merker) per calcolare risultati logici.
5. **Compiti di sistema:** autotest, comunicazione e gestione eventi. Poi il ciclo ricomincia dal punto 1.

 Il tempo necessario a completare questa sequenza è il **tempo di ciclo** del PLC.

---

## 0.3.2 Funzionamento sequenziale e tempi

Il PLC elabora le istruzioni **una dopo l'altra**, e questo porta a due effetti principali:

- **Tempo di esecuzione delle istruzioni:** varia in base al tipo di istruzione. Ad esempio, leggere un ingresso può richiedere un solo microsecondo.
- **Possibili contraddizioni:** istruzioni che producono risultati incompatibili possono generare incertezze già in fase di programmazione.

### 0.3.2.1 Tipologie di tempi

- **Tempo di esecuzione:** tempo tra la lettura di un'istruzione e l'inizio della lettura della successiva.
- **Tempo di ciclo (o di scansione):** tempo necessario a completare tutte le istruzioni del programma.

 Diversi costruttori implementano modalità diverse per **acquisizione degli ingressi e aggiornamento delle uscite**, creando tre principali tipologie di cicli macchina.

---

### 0.3.2.2 Immagini di processo

- Gli I/O digitali e analogici locali vengono aggiornati ciclicamente in memoria tramite l'immagine di processo.
  - L'immagine riflette lo stato reale di ingressi e uscite fisici (CPU, moduli I/O, signal board).
  - Durante l'esecuzione del programma utente, la CPU può leggere o scrivere direttamente sugli I/O fisici aggiungendo il suffisso :P agli indirizzi, bypassando l'immagine di processo.
- 

### Note pratiche

- Il ciclo di elaborazione **non ha durata costante**.
- Eventi molto rapidi possono essere persi tra un ciclo e l'altro.
- Il tempo di ciclo rappresenta un **limite alla velocità di risposta** a un allarme.
- Il funzionamento sequenziale e l'uso delle immagini di processo garantiscono **coerenza logica** e stabilità nelle uscite durante l'esecuzione del programma.

## 0.4 Linguaggi di programmazione

I PLC richiedono linguaggi specifici. La normativa IEC 61131-3 definisce cinque linguaggi standard: tre grafici e due testuali.

Linguaggi grafici	Linguaggi testuali
Diagramma funzionale sequenziale (SFC, Sequential Functional Chart)	Lista di istruzioni (IL, Instruction List)
Linguaggio a contatti (LD, Ladder Diagram)	Testo strutturato (SCL, Structured Control Language)
Diagramma a blocchi funzionali (FBD, Function Block Diagram)	-

## 0.5 Confronto PC-PLC

Un PC può svolgere funzioni simili a un PLC se interfacciato correttamente, ma:

- I PLC sono progettati per ambienti industriali, resistenti a disturbi elettromagnetici e condizioni estreme.
- I PC sono più dinamici e flessibili, ma meno adatti a impianti industriali senza software specializzato.

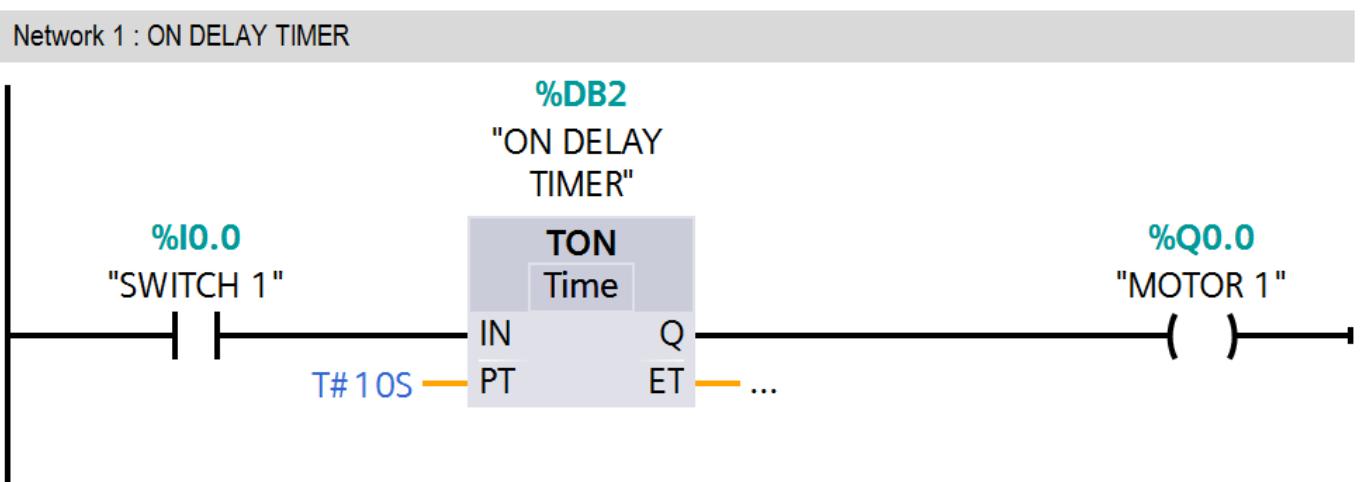
- I PLC sono utilizzabili anche da personale non esperto di informatica, grazie ai linguaggi dedicati.

# 1. BLOCCHI - FUNZIONI - OPERATORI

## 1.1 Temporizzatori (Timers)

[TIA Portal Temporizzatori](#)

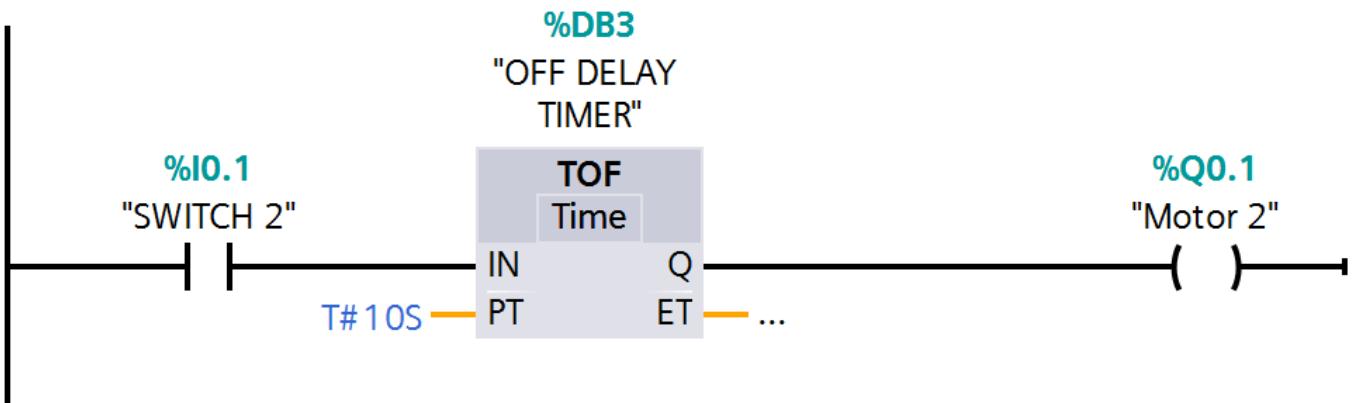
### 1.1.1 TON – Timer On Delay



Campo	Tipo	Descrizione
IN	BOOL	Segnale di abilitazione del timer.
PT	TIME	Tempo di preset (es. T#10s ).
Q	BOOL	Uscita: diventa vera dopo che il tempo PT è trascorso con IN = 1.
ET	TIME	Tempo effettivo trascorso.
Descrizione:		Il TON ritarda l'attivazione dell'uscita. Se IN è attivo finché ET raggiunge PT, allora Q diventa TRUE.

## 1.1.2 TOF – Timer Off Delay

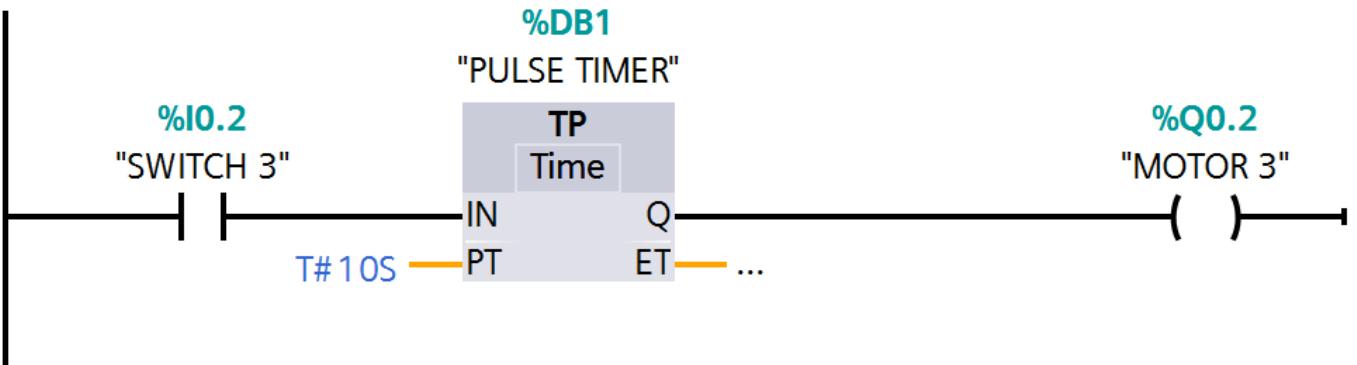
Network 2 : OFF DELAY TIMER



Campo	Tipo	Descrizione
IN	BOOL	Segnale di ingresso.
PT	TIME	Tempo di mantenimento in uscita dopo che IN torna FALSE (es. T#10s ).
Q	BOOL	Uscita che rimane vera per PT dopo che IN è passato a 0.
ET	TIME	Tempo effettivo di ritardo.
<b>Descrizione:</b> Mantiene l'uscita attiva per un tempo prestabilito dopo lo spegnimento dell'ingresso.		

## 1.1.3 TP – Timer Pulse

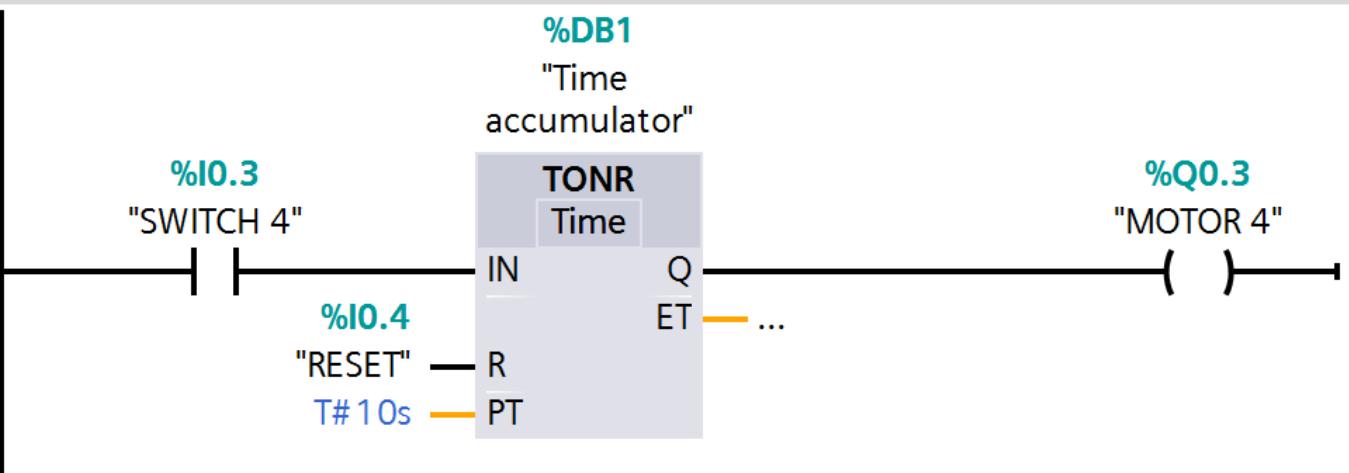
### Network 3 : Pulse Timer



Campo	Tipo	Descrizione
IN	BOOL	Impulso di attivazione (fronte di salita).
PT	TIME	Durata dell'impulso.(es. T#10s ).
Q	BOOL	Uscita attiva per PT al fronte di salita di IN.
ET	TIME	Tempo trascorso dall'attivazione.
<b>Descrizione:</b> Genera un impulso temporizzato ogni volta che IN passa da 0→1.		

### 1.1.4 TONR – Timer On Delay Retentive (Retentivo)

#### Network 4 : Time Accumulator

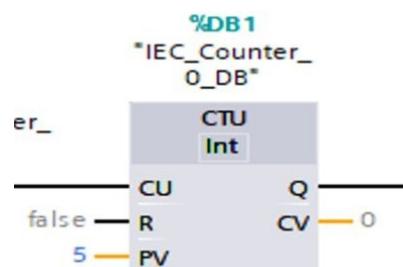


Campo	Tipo	Descrizione
IN	BOOL	Segnale di conteggio/attivazione.
PT	TIME	Tempo di preset (es. T#10s ).
Q	BOOL	Vero quando ET $\geq$ PT.
ET	TIME	Tempo accumulato (non azzera se IN torna 0).
R	BOOL	Reset del timer.
<b>Descrizione:</b> Simile al TON, ma conserva il valore ET anche se IN torna a 0 (fino al reset) — utile quando la temporizzazione deve essere “ritentiva”.		

## 1.2 Contatori (Counters)

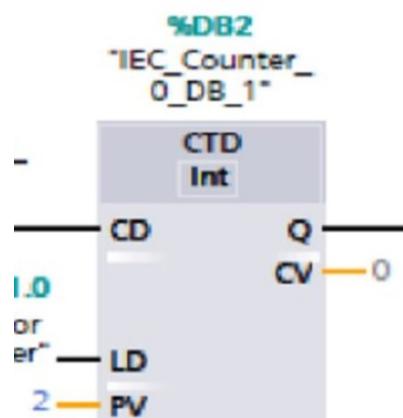
[TIA Portal Contatori](#)

### 1.2.1 CTU – Counter Up



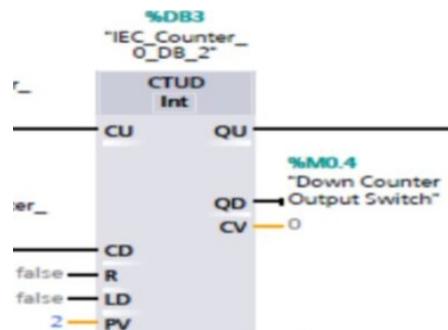
Campo	Tipo	Descrizione
CU	BOOL	Fronte di salita → incremento del contatore.
R	BOOL	Reset conteggio.
PV	INT / DINT	Valore di preset.
Q	BOOL	Uscita vera se CV $\geq$ PV.
CV	INT / DINT	Valore corrente del conteggio.
<b>Descrizione:</b> Incrementa il conteggio ad ogni fronte di salita su CU. Quando CV raggiunge o supera PV, Q diventa TRUE.		

## 1.2.2 CTD – Counter Down



Campo	Tipo	Descrizione
CD	BOOL	Fronte di salita → decremento del contatore.
LD	BOOL	Carica il valore iniziale (PV).
PV	INT / DINT	Valore iniziale.
Q	BOOL	Vera se CV = 0.
CV	INT / DINT	Valore attuale del conteggio.
<b>Descrizione:</b> Conta verso il basso ad ogni fronte di salita su CD; quando CV arriva a 0, Q diventa TRUE.		

### 1.2.3 CTUD – Counter Up/Down



Campo	Tipo	Descrizione
CU	BOOL	Conteggio verso l'alto.
CD	BOOL	Conteggio verso il basso.
R	BOOL	Reset conteggio.
LD	BOOL	Carica valore PV.
PV	INT / DINT	Valore di preset.
QU	BOOL	Attivo se CV $\geq$ PV.
QD	BOOL	Attivo se CV $\leq$ 0.
CV	INT / DINT	Valore attuale del conteggio.
Descrizione: Combina funzioni di contatore up e down, permettendo incremento o decremento a seconda degli ingressi.		

## 1.3 Comparatori e Operatori Logici

[TIA Portal Comparatori](#)

Nome	Ingressi	Tipo	Funzione
EQ	IN1, IN2	INT / REAL	Vera se IN1 = IN2
NE	IN1, IN2	INT / REAL	Vera se IN1 ≠ IN2
GT	IN1, IN2	INT / REAL	Vera se IN1 > IN2
GE	IN1, IN2	INT / REAL	Vera se IN1 ≥ IN2
LT	IN1, IN2	INT / REAL	Vera se IN1 < IN2
LE	IN1, IN2	INT / REAL	Vera se IN1 ≤ IN2
<b>Descrizione:</b> Comparatori numerici standard, usati per confrontare variabili in logica di controllo.			
Operatori logici standard come AND, OR, XOR, NOT gestiscono variabili BOOL e combinazioni logiche.			

## 1.4 Blocchi Aritmetici

[TIA Portal Funzioni matematiche](#)

Nome	Ingressi	Uscita	Tipo	Descrizione
ADD	IN1, IN2	OUT	INT / REAL	Somma di IN1 e IN2
SUB	IN1, IN2	OUT	INT / REAL	Sottrazione: IN1 - IN2
MUL	IN1, IN2	OUT	INT / REAL	Moltiplicazione
DIV	IN1, IN2	OUT	REAL	Divisione
MOD	IN1, IN2	OUT	INT	Resto della divisione
ABS	IN	OUT	INT / REAL	Valore assoluto
SQRT	IN	OUT	REAL	Radice quadrata
<b>Descrizione:</b> Blocchi per operazioni aritmetiche, utili nei controlli analogici o nei calcoli ausiliari.				

## 1.5 Blocchi di Memoria e Gestione Bit

### 1.5.1 SR – Set / Reset

[TIA Portal Istruzioni di base con i bit](#)

Campo	Tipo	Descrizione
S	BOOL	Imposta l'uscita a 1.
R	BOOL	Resetta l'uscita a 0.
Q	BOOL	Uscita memorizzata.
<b>Descrizione:</b> Latch logico che mantiene lo stato fino a reset.		

---

## 1.5.2 RS – Reset / Set

Campo	Tipo	Descrizione
R	BOOL	Reset prima di tutto.
S	BOOL	Setta l'uscita a 1.
Q	BOOL	Uscita memorizzata.
<b>Descrizione:</b> Variante del SR con priorità invertita tra Reset e Set.		

---

## 1.6 Blocchi di Trigger

Nome	Ingresso	Tipo	Funzione
R_TRIGGER	CLK	BOOL	Uscita TRUE solo al fronte di salita ( $0 \rightarrow 1$ ).
F_TRIGGER	CLK	BOOL	Uscita TRUE solo al fronte di discesa ( $1 \rightarrow 0$ ).
<b>Descrizione:</b> Utili per rilevare cambiamenti di stato (fronti) e attivare logiche solo al passaggio.			

---

## 1.7 Blocchi di Trasferimento / Conversione

Nome	Ingressi	Tipo	Descrizione
MOV	IN	Qualunque	Copia il valore di IN in OUT
SWAP	IN	BYTE / WORD	Inverte l'ordine dei byte
INT_TO_REAL	IN	INT → REAL	Converte intero in reale
REAL_TO_INT	IN	REAL → INT	Converte reale in intero
SEL	G, IN0, IN1	BOOL, Qualunque	Se G = TRUE, OUT = IN1 altrimenti IN0
<b>Descrizione:</b> Blocchi utili per gestire dati, tipi, condizioni di selezione, conversioni nei programmi PLC.			

## 1.8 Blocchi di Regolazione

Nome	Ingressi	Tipo	Descrizione
PID	PV, SP, Kp, Ki, Kd	REAL	Regolatore proporzionale-integrale-derivativo
LIMIT	MIN, IN, MAX	REAL	Limita IN tra MIN e MAX
SCALE	IN, IN_MIN, IN_MAX, OUT_MIN, OUT_MAX	REAL	Scala un valore da un range all'altro
<b>Descrizione:</b> Utilizzati nei processi analogici o nei controlli che richiedono regolazione automatica.			

## 1.9 Blocchi per Stringhe

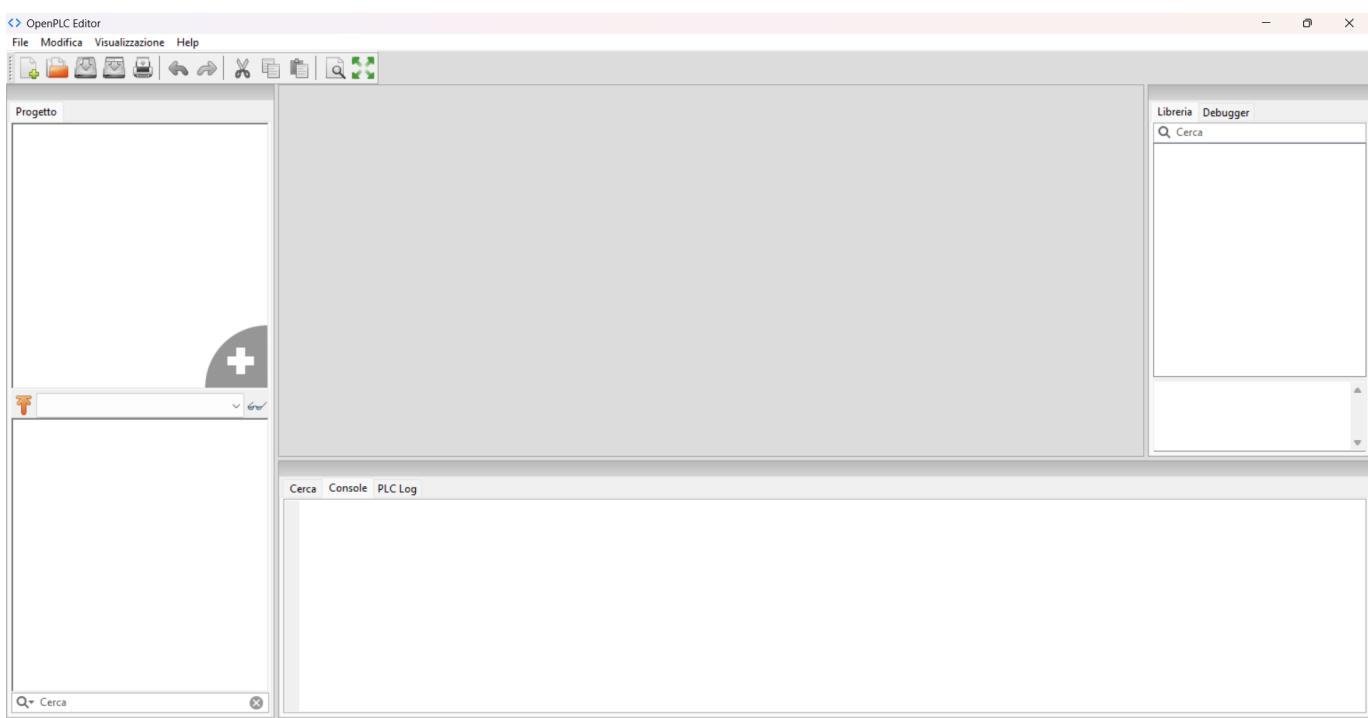
Nome	Ingressi	Tipo	Funzione
CONCAT	IN1, IN2	STRING	Concatena due stringhe
LEN	IN	STRING	Restituisce la lunghezza della stringa
LEFT / RIGHT / MID	IN, N, [LEN]	STRING / INT	Estrae porzioni di testo
<b>Descrizione:</b> Utili nei programmi PLC moderni per la gestione di testi, messaggi, visualizzazioni su pannello HMI.			

## Note di utilizzo

- I tipi di variabile **BOOL, TIME, INT, REAL, STRING** ecc. sono definiti dallo standard IEC 61131-3.
- Quando vengono inseriti questi blocchi nei linguaggi PLC (LD, FBD, ST) devi collegare correttamente i tipi di variabile ai pin indicati.

## 2. OPEN PLC EDITOR - GUIDA AL PROGRAMMA

### 2.1 Primo avvio



All'avvio del programma si incontra la schermata comune per tutti gli ambienti di sviluppo.

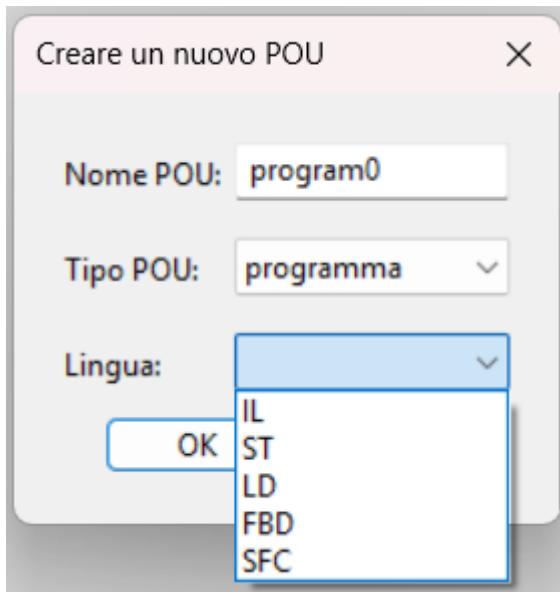
Per generare un nuovo progetto con la relativa scelta di parametri:



- File -> **Nuovo** o, alternativamente, sull'icona 
- Si rende necessario selezionare una cartella di destinazione **VUOTA** dove verranno salvati tutti i file appartenenti al progetto sviluppato.

Una volta selezionata la cartella apparirà un menu contestuale dove inserire:

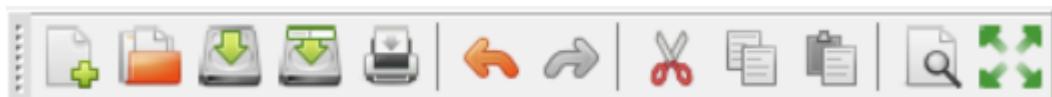
- Il nome del programma (Es. "Cancello\_Automatico") ;
- Il tipo di POU (Program Organization Units) ;
- Il tipo di linguaggio tra **LD** - ladder, **ST** - testo strutturato, **FBD** - diagramma a blocchi funzionali e **SFC** - diagramma sequenziale funzionale.



## 2.2 Il primo programma ladder

### 2.2.1 L'interfaccia e i diversi ambienti

In alto si trovano le 3 barre multifunzione principali:



- La prima barra viene usata per le funzioni di salvataggio, stampa, "Undo / Redo", Taglia/Copia/Incolla dagli appunti, funzione "cerca nel progetto" e infine per attivare / disattivare la modalità a schermo intero.



- La seconda barra viene utilizzata per le funzioni di **Avvio simulazione PLC**, **Generare programma per OpenPLC Runtime** ( maggiori dettagli al capitolo "3. OPEN PLC RUNTIME

- GUIDA AL PROGRAMMA" ), Trasferire programma al PLC e avviare il debug live di un plc da remoto.



- La terza barra contiene tutti i comandi necessari alla compilazione di un programma nel tipo di linguaggio selezionato precedentemente, in questo caso Ladder.

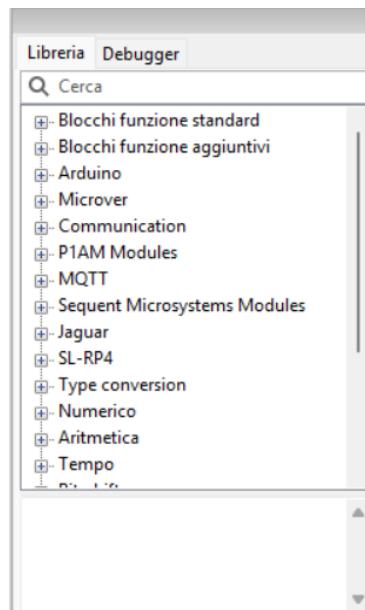
Viene utilizzata per le funzioni di **Selezione oggetti**, **Spostare la vista**, **Aggiungere commenti**, **Inserire barre di alimentazione**, **Inserire relè**, **Inserire contatti**, **Creare nuove variabili**, **Creare nuovi blocchi**, **Creare nuove connessioni**.

---

**Nota :** Per inserire il tipo di blocco selezionato all'interno dell'area di progettazione è necessario selezionare il tipo di comando dalla barra (o dalla libreria) e poi cliccare all'interno dell'area di progettazione stessa per posizionarlo.

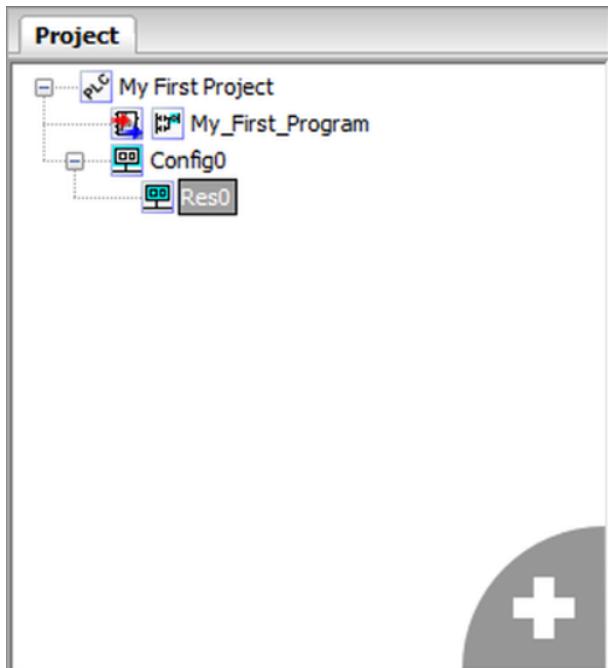
---

Oltre che poter selezionare il tipo di blocco da inserire dalla barra multifunzione è possibile utilizzare anche la libreria, collocata nella parte destra dello schermo e riportata di seguito.



## 2.2.2 Variabili Globali e tempo ciclo

Una volta creato il programma si può notare che sono state create in automatico una voce di configurazione e una di risorsa :



Per cambiare queste impostazioni è sufficiente cliccare sulla dicitura " Res " per accedere al menu contestuale:

La finestra principale mostrerà un **primo menu** per la creazione delle variabili globali che utilizzeremo poi nel programma, un **secondo menu** per la gestione dei programmi stessi ed un **terzo menu** per gestire le diverse istanze che potrebbero coesistere contemporaneamente.

- 
- Per sapere come inserire le variabili all'interno del **primo menu** fare riferimento a "2.3 Notazione Input & Output".
-

Nel **secondo menu** inerente alla gestione dei programmi bisogna prestare molta attenzione al valore presente all'interno del campo " Intervallo ".

Ogni programma PLC funziona ciclicamente: viene eseguita la prima istruzione fino ad arrivare all'ultima in maniera sequenziale per poi ripartire da capo.

E' fondamentale nella scelta dell'intervallo di scansione prendere in considerazione dei parametri:

1. Il programma viene eseguito su Windows o Linux tramite OpenPLC Runtime?

Se il programma plc viene eseguito su un sistema con un OS ( e conseguente kernel ) preesistente allora è bene non alterare eccessivamente il tempo ciclo rispetto il valore default di 20 ms poichè ciò potrebbe comportare una serie di conflitti e portare ad una esecuzione non affidabile.

2. Il programma viene eseguito su piattaforme a basso coding ( Arduino/ESP32 ) ?

Se il programma viene esportato da OpenPLC Runtime per essere caricato su di una board a basso coding ciò comporta due cose: il tempo ciclo sarà affidabile ma allo stesso tempo, data la poca potenza di calcolo di queste schede, la piattaforma sarà maggiormente suscettibile anche a minimi aumenti di carico computazionale dato dalla diminuzione del tempo di ciclo.

Per i motivi citati sopra è consigliabile prevedere un periodo di live debug immediatamente dopo al caricamento del programma per assicurarsi del normale funzionamento delle operazioni attese.

## 2.3 Notazione Input & Output

---

Le applicazioni PLC interagiscono con il mondo esterno tramite moduli INPUT e OUTPUT e/o protocolli di comunicazioni SCADA.

Durante la progettazione dell'applicazione PLC si rende necessario decidere quali variabili verranno usate dal programma stesso per assegnare gli indirizzi di input/output necessari.

OpenPLC utilizza la nomenclatura standard IEC 61131-3 per l'assegnazione di INPUT, OUTPUT e LOCAZIONI DI MEMORIA.

L'assegnazione avviene tramite l'utilizzo di sequenze di caratteri speciali così composte:

/	Pref. di locaz.	Pref. di dimensione	Indirizzo fisico
%	I , Q , M	X , B , W , D , L	X.Y

### 2.3.1 Prefisso di locazione

- " I " = Locazione di una variabile di **INPUT** ;
- " Q " = Locazione di una variabile di **OUTPUT** ;
- " M " = Locazione di una variabile di **MEMORIA** .

### 2.3.2 Prefisso di dimensione

Prima di elencare i vari prefissi per indicare le allocazioni di memoria è importante ricordare la definizione di **bit** : il bit corrisponde alla più piccola unità quando si parla di misure informatiche.

Rappresenta la parte più piccola in assoluto in una informazione e può assumere solamente due valori : 0 / 1 .

- " X " = Dimensione pari a 1 bit ;
- " B " = Dimensione pari a 1 byte (8 bits) ;
- " W " = Dimensione pari a 1 word (16 bits) ;
- " D " = Dimensione pari a 1 double word (32 bits) ;
- " L " = Dimensione pari a 1 long word (64 bits) ;

### 2.3.3 Esempio di lettura variabile

Poniamo come esempio di voler leggere lo stato di una variabile di tipo Bool ( variabile che può assumere valore 0 / 1 ).

La dichiarazione seguirà la sintassi mostrata precedentemente e sarà così composta:

- %IX0.0 = variabile di **input** all'ingresso fisico **0.0** di dimensione pari a **1 bit** (0/1) ;
- %QX0.1 = variabile di **output** all'uscita fisica **0.1** di dimensione pari a **1 bit** (0/1) ;

---

**Nota :** In OpenPLC la mappatura degli I/O è dipendente dalla piattaforma utilizzata, per maggiori dettagli in merito all'assegnazione delle stesse all'interno di OpenPLC fare

riferimento al Paragrafo "[3. OPEN PLC RUNTIME - GUIDA AL PROGRAMMA](#)" .

---

**Nota :** Gli indirizzi di tipo **bit** (X) utilizzati nel **PLC** hanno una gerarchia precisa tra le due componenti dell'indirizzo stesso:

- La parte dell'indirizzo più a destra ha l'importanza minore e può avere un valore numerico da 0 a 7. Questo numero rappresenta una posizione in un byte.

Sostanzialmente ogni cifra ( da 0 a 7 ) dopo il punto che separa le due componenti dell'indirizzo corrisponde ad un bit che può assumere un valore pari a 0 o 1.

- La parte dell'indirizzo più a sinistra ha l'importanza maggiore e può avere un valore numerico da 0 a 1023. Questi 1024 valori rappresentano i 1024 byte che a loro volta possono avere 8 bit dal singolo valore di 0/1.
- 

**Nota :** Le dimensioni per memorie diverse da "X" hanno una sola cifra nel loro indirizzo.

Tali memorie non devono avere un punto che divide le due cifre e devono soprattutto essere in quantità inferiore rispetto la dotazione della piattaforma in utilizzo

---

### 2.3.4 Esempi di indicizzazioni non valide

1. **%IX0.8** = indirizzo non valido poichè la cifra di indirizzo secondaria è maggiori di 7 ;
2. **%QX0.0.1** = indirizzo non valido poichè la gerarchia non può essere composta da un indirizzo a 3 cifre (X.Y oppure X sono le uniche due nomenclature ammesse dal programma ) ;
3. **%IB1.1** = indirizzo non valido perchè l'unico con il quale è ammesso una indicizzazione composta da due cifre è l'indirizzo di tipo "X" .

## 3. Il primo programma in OpenPLC - guida passo - passo

---

### [Open PLC Editor - creare il primo programma](#)

Ora che hai imparato le basi di OpenPLC, è il momento di **mettere le mani in pasta** e scrivere un po' di codice con **OpenPLC Editor**.

Questo progetto è un semplice **interruttore on/off**. Ti serviranno:

- Un dispositivo con OpenPLC Runtime installato
- Due pulsanti
- Un LED

Prima che tu lo chieda: se stai eseguendo OpenPLC su una di quelle **schede industriali** (come UniPi, PiXtend, ecc.), potresti voler sostituire il LED con una **lampada industriale a 24V**. Funzionerà nello stesso identico modo.

Per iniziare, collega il tuo circuito in questo modo:



#### Osservazioni importanti:

- +V rappresenta il livello di tensione positivo del tuo dispositivo. Ad esempio, per le **schede Arduino** è solitamente +5V, per il **Raspberry Pi** è +3,3V, mentre per le **schede industriali** è +24V.
- **PB1** e **PB2** sono pulsanti (nel caso l'immagine non fosse abbastanza chiara). **R1** e **R2** sono resistenze di **pull-down**: qualsiasi valore compreso tra **1KΩ** e **10KΩ** andrà bene.

Se stai utilizzando una **scheda industriale** o una **scheda con resistenze di pull-down interne**, molto probabilmente **non avrai bisogno di R1 e R2 nel tuo circuito**.  
In tal caso puoi collegare **PB1** e **PB2** direttamente a **%IX0.0** e **%IX0.1**.

- Sulle **schede Raspberry Pi**, i primi due ingressi (**%IX0.0** e **%IX0.1**) sono **invertiti a livello hardware**.  
Questo può causare problemi, poiché sembrerà che i pulsanti siano **sempre premuti**.  
Puoi risolvere il problema **invertendo il segnale di ingresso** nel programma PLC (utilizzando contatti negati, se sai come farlo) oppure semplicemente **usando altri ingressi**, ad esempio **%IX0.2** e **%IX0.3**.

Inizia creando un **nuovo progetto** in **OpenPLC Editor**.

Per farlo, ti basta cliccare su **File → New**.



Apparirà una finestra di salvataggio che ti permetterà di scegliere **dove salvare il tuo progetto**. I progetti di **OpenPLC Editor** non sono singoli file, ma **cartelle** che contengono tutti i dati del progetto.

Non puoi salvare un progetto in una cartella che contiene già altri file.

Crea quindi una **nuova cartella**, aprila e selezionala come **posizione di destinazione** per il tuo progetto.

Una volta scelta la posizione, **OpenPLC Editor** creerà il progetto con le **impostazioni e configurazioni predefinite**, e aprirà una nuova finestra che ti chiederà di creare un nuovo **POU**.

**POU** significa *Program Organization Unit* ed è l'unità in cui viene memorizzato tutto il codice che scriverai nel tuo progetto.

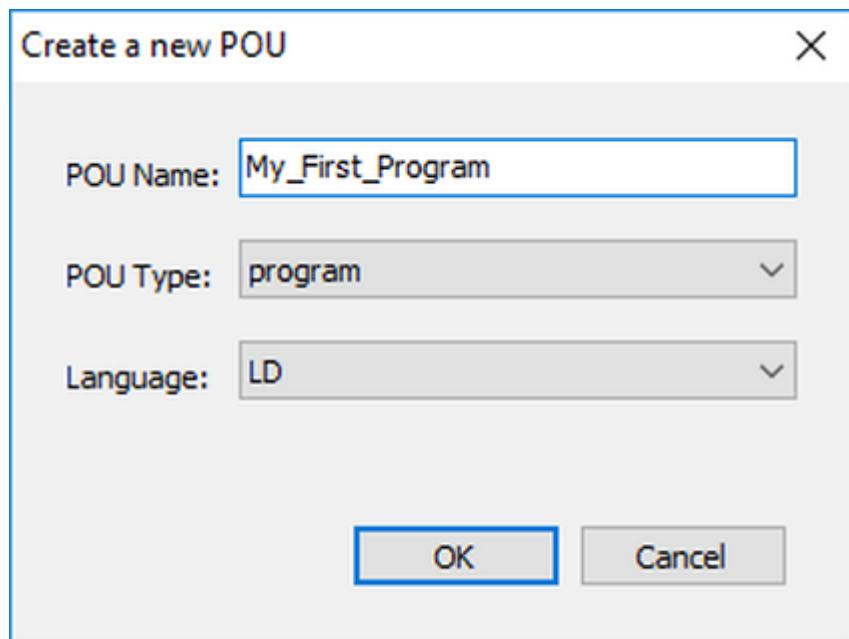
Esistono tre tipi di POU che puoi creare:

1. **Program** – codice applicativo che combina ingressi, uscite, funzioni e blocchi funzione.
2. **Function** – codice riutilizzabile che restituisce un valore.
3. **Function Block** – codice riutilizzabile che può mantenere uno stato (istanza).

Per questo tutorial creeremo solo un POU di tipo “**Program**”.

Inserisci quindi un **nome per il tuo programma**, assicurati che il **POU Type** sia impostato su “**program**” e che il **Language** sia “**LD**” (*Ladder Diagram*).

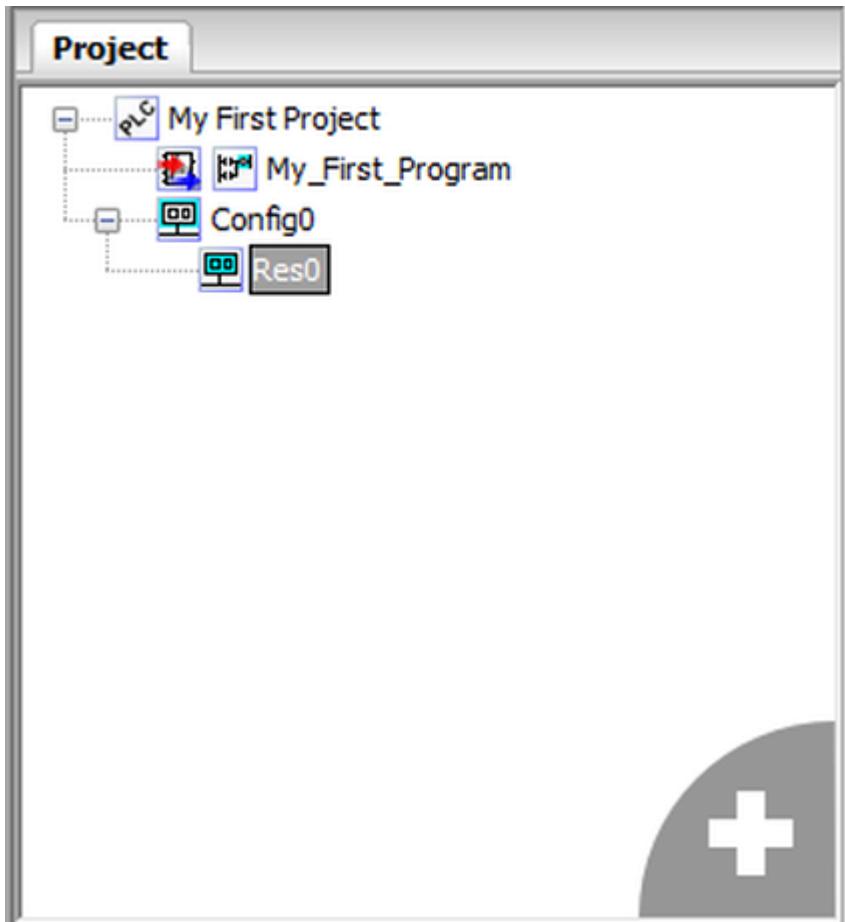
Ricorda inoltre che il **nome del programma** non può contenere spazi o caratteri speciali.



Quando crei un nuovo programma, **OpenPLC Editor** genera automaticamente per te una **configurazione**, una **risorsa**, un **task** e un’**istanza**.

Questi elementi servono a indicare a **OpenPLC** come gestire il tuo programma (ad esempio: quando richiamare una funzione, come eseguire il ciclo operativo, ecc.).

Puoi modificare questi elementi facendo doppio clic su “**Res0**” nel pannello di sinistra.



La finestra principale mostra, nella parte superiore, un campo per l'inserimento delle variabili globali (che ti permette di creare variabili accessibili in tutto il programma), oltre a una finestra Tasks e una finestra Instances.

Puoi creare nuovi task cliccando sul segno "+" verde all'interno della finestra Tasks.

In questo progetto **non creeremo nuovi task**, ma potresti voler modificare l'**intervallo** del task in base all'hardware su cui stai eseguendo OpenPLC.

I programmi PLC sono **ciclici**, il che significa che:

1. Iniziano dalla **prima istruzione**;
2. Proseguono fino all'**ultima**;
3. **Attendono** un breve intervallo;
4. E **ricominciano** da capo.

Il parametro **Interval** definisce **ogni quanto tempo** il ciclo del programma viene eseguito.

Il valore predefinito è **20 ms**, cioè il programma viene eseguito **una volta ogni 20 millisecondi**.

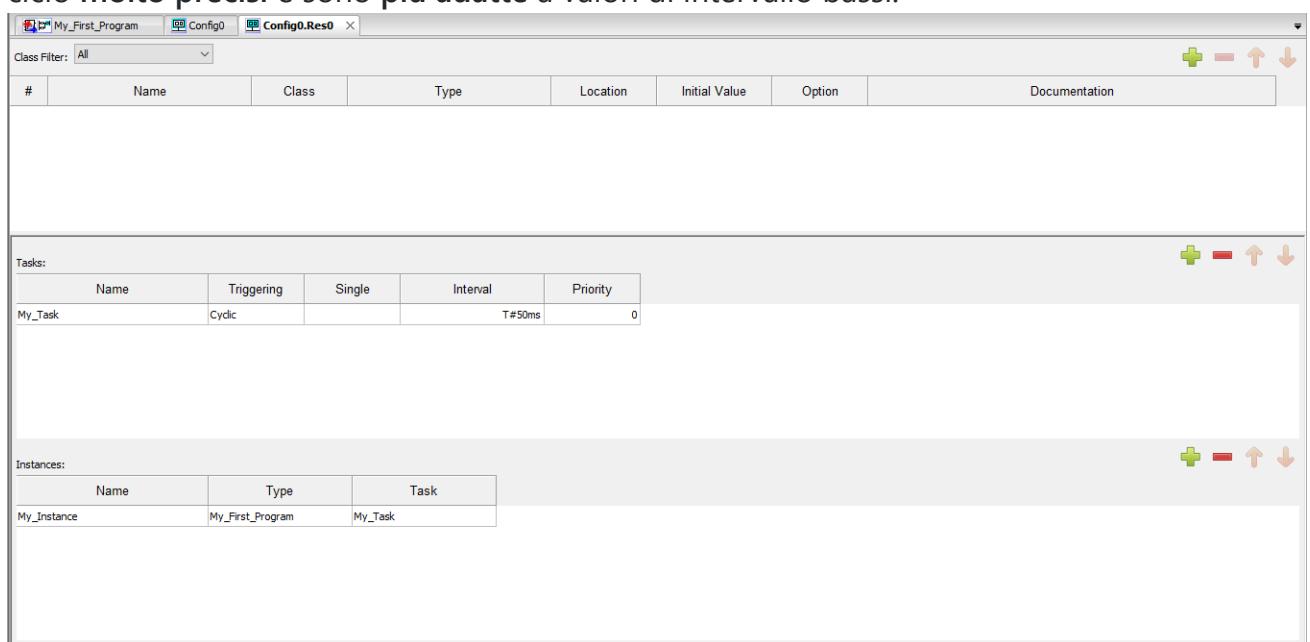
Se vuoi che il programma venga eseguito più spesso, puoi ridurre questo tempo.

Tuttavia, se scegli un intervallo troppo basso (ad esempio **1 ms**), il tuo programma potrebbe

consumare il 100% della CPU del dispositivo e comunque non riuscire a mantenere il ciclo stabile.

⌚⌚⌚ Un valore sicuro per la maggior parte delle piattaforme è **20 ms**.

- Le piattaforme con **sistemi operativi** (come Windows o Linux) sono **meno reattive** e possono comportarsi in modo imprevedibile con cicli troppo brevi, poiché il sistema operativo può interferire con la pianificazione dei cicli PLC in base alle **priorità del kernel**.
- Le piattaforme **bare metal**, come le **schede Arduino**, invece, riescono a mantenere tempi di ciclo **molto precisi** e sono **più adatte** a valori di intervallo bassi.



Ora che il progetto è finalmente stato creato, puoi iniziare a **disegnare il diagramma a logica Ladder**.

Fai clic sul **nome del tuo programma** nel pannello di sinistra per aprire l'**editor Ladder**. La parte superiore della schermata è riservata alle **variabili**, mentre la parte **centrale** viene utilizzata per il **diagramma logico**.

Cominciamo aggiungendo alcune variabili:

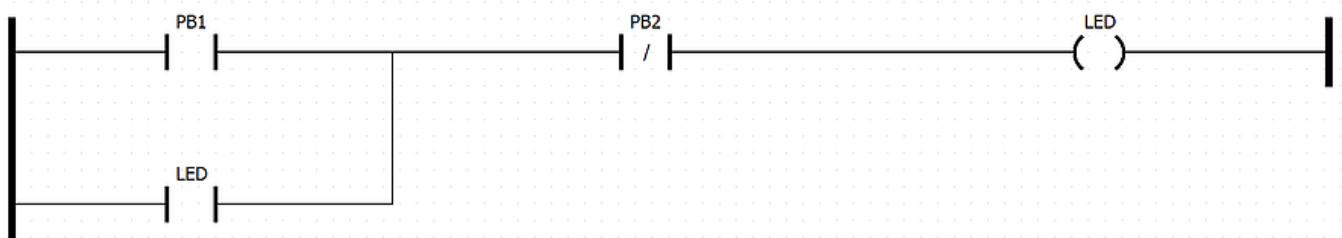
clicca sul segno “+” verde e aggiungi **tre variabili**.

Nome	Classe	Tipo	Posizione
PB1	Local	BOOL	%IX0.0
PB2	Local	BOOL	%IX0.1
LAMP	Local	BOOL	%QX0.0

#	Name	Class	Type	Location	Initial Value	Option	Documentation
1	PB1	Local	BOOL	%IX0.0			
2	PB2	Local	BOOL	%IX0.1			
3	LED	Local	BOOL	%QX0.0			

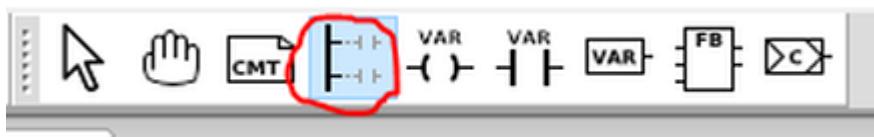
Quello che vogliamo ottenere con questo programma è che, **ogni volta che PB1 viene premuto, la LAMP si accenda e rimanga accesa fino a quando non viene premuto PB2.**

Questo comportamento si realizza tramite un **semplice circuito di mantenimento (latch)** nella logica Ladder, come mostrato di seguito:



Per creare questo circuito nell'editor, **inizia aggiungendo la barra di alimentazione sinistra (left power rail).**

Puoi farlo **cliccando sull'icona della barra di alimentazione** nella **barra degli strumenti**.



Regola il numero di pin della **barra di alimentazione sinistra** a **50** – questo è un valore ragionevole se desideri aggiungere altri gradini (rungs) al tuo programma in seguito.

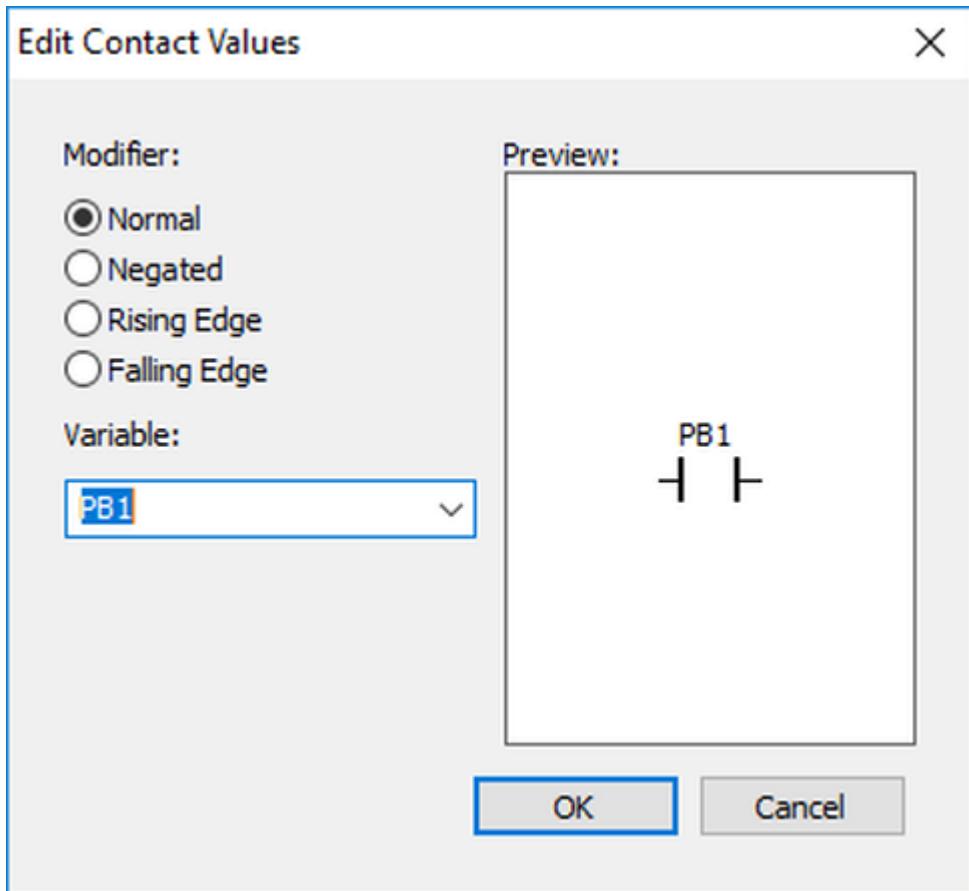
Aggiungi un'altra **barra di alimentazione** con conteggio **50**, ma questa volta seleziona “right power rail” nelle proprietà. Posizionala sul lato destro dello schermo.

Questo è sufficiente per impostare i **rungs** del tuo **Diagramma Ladder**.

Ora puoi iniziare ad aggiungere gli **elementi del diagramma ladder**.

Aggiungi un **contatto** cliccando sul pulsante **\*contact\*** nella barra degli strumenti oppure cliccando con il tasto destro nella finestra vuota dell'editor e selezionando **Add → Contact**.

Nella finestra che appare, sotto il parametro **“Variable”**, seleziona **PB1** per associare il nuovo contatto alla variabile **PB1**

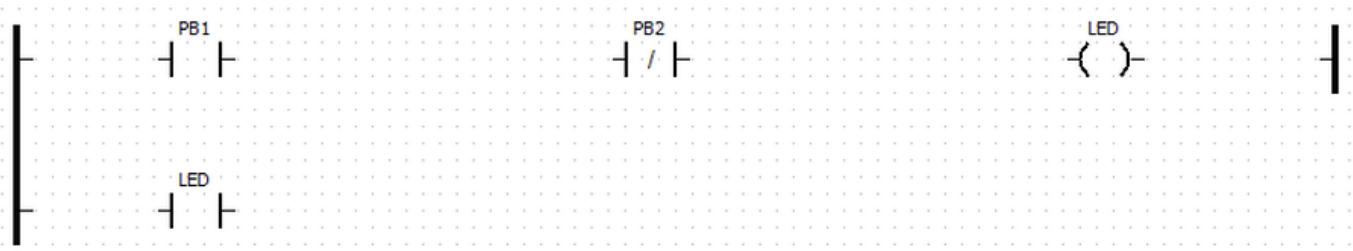


Ripeti il processo per aggiungere **altri due contatti**, uno associato a **PB2** e un altro associato a **LAMP**.

Per il contatto **PB2**, seleziona **Negated** come modificatore.

Infine, aggiungi una **bobina (coil)** cliccando sul pulsante *coil* nella barra degli strumenti oppure cliccando con il tasto destro nella finestra vuota dell'editor e selezionando **Add → Coil**.  
Associa la nuova bobina alla variabile **LAMP** e aggiungi una **barra di alimentazione destra** per chiudere il circuito.

Posiziona i componenti in modo che appaiano come nell'immagine seguente:



Il passaggio finale consiste nel **collegare tutti i componenti** trascinando le loro estremità per formare una linea.

Collega il lato **sinistro** dei contatti **PB1** e **LED** alla **barra di alimentazione sinistra**.

Collega il **lato destro** di **PB1** a **PB2**, il **lato destro** di **PB2** alla **bobina LED**, e il **lato destro** della

bobina LED alla barra di alimentazione destra.

Disegna il circuito parallelo del contatto LED collegando il lato destro del contatto LED al lato sinistro di PB2.

Il tuo progetto finale dovrebbe apparire come la **prima immagine** del diagramma a scala mostrata in questo tutorial.

Questo circuito inizialmente ha la **lampada (LED) spenta**.

Quando premi PB1, anche solo per un istante, il circuito **accende la LAMPADA** (purché PB2 non sia premuto).

Una volta accesa, la **LAMPADA bypassa il pulsante PB1**, mantenendosi accesa **anche dopo aver rilasciato PB1**.

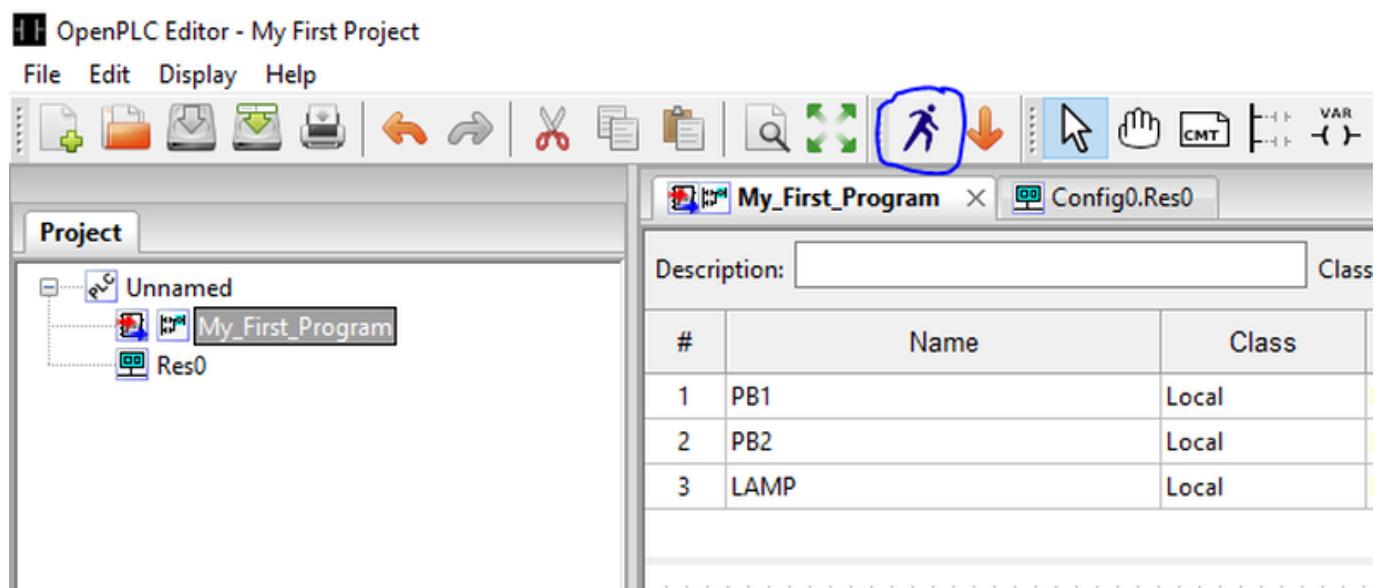
Questo è un piccolo trucco nella **logica a scala (ladder logic)**: è infatti possibile **utilizzare le uscite come contatti**.

L'unico modo per spegnere la **LAMP** è premere **PB2**.

Poiché **PB2** è un **contatto negato**, aprirà il circuito una volta premuto, spegnendo così la lampada.

Ora che il progetto è stato creato è un buon momento per **testarlo prima di caricarlo sull'OpenPLC Runtime**.

Risulta possibile simulare il comportamento del programma cliccando su **Start PLC Simulation** nella barra degli strumenti.

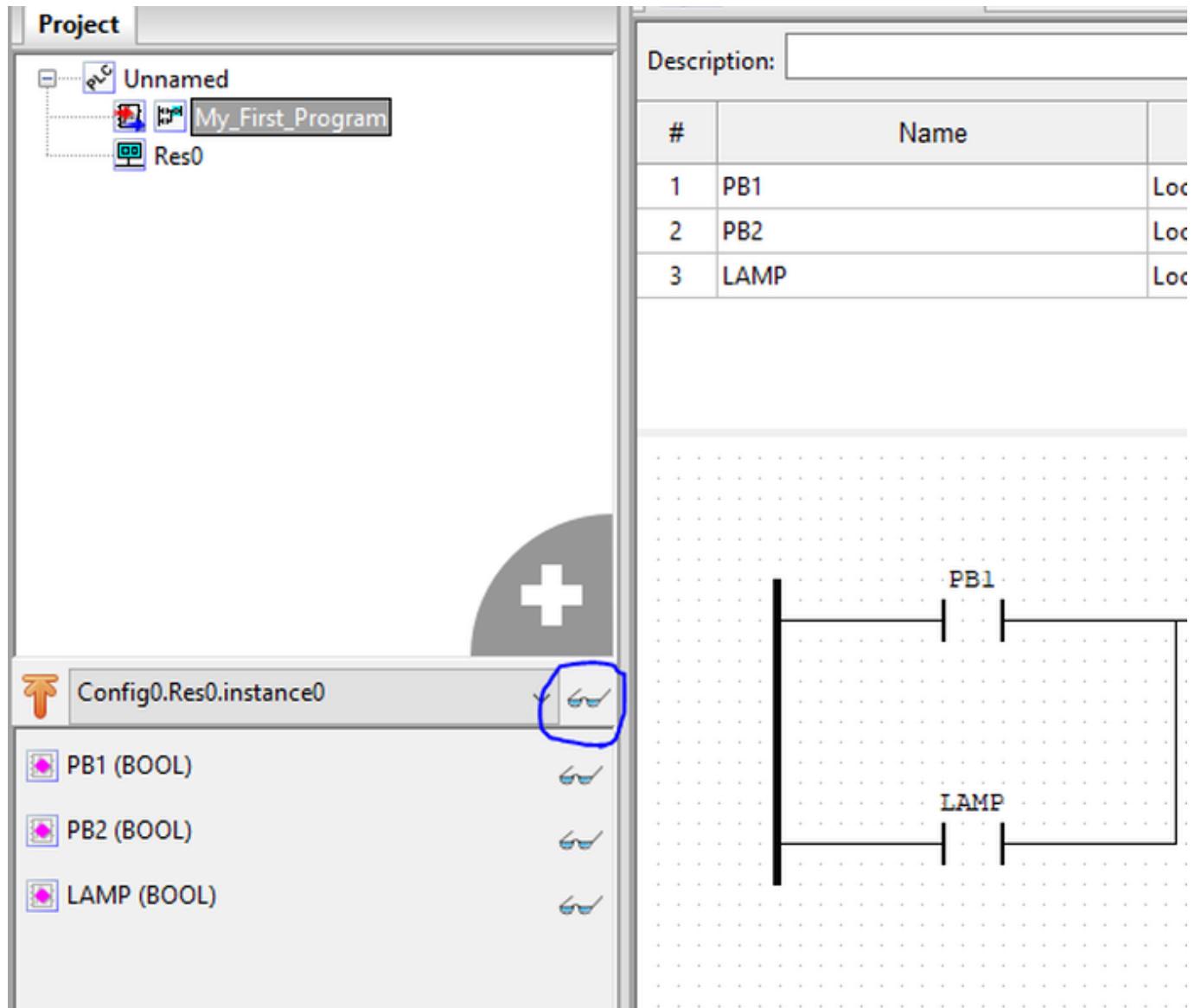


Quando clicchi su quel pulsante, il tuo progetto viene **compilato** utilizzando un processo simile a quello dell'**OpenPLC Runtime**.

Se ci sono **errori nel programma**, la compilazione **fallirà** e verrai avvisato degli errori nel **pannello Console** situato nella parte inferiore dello schermo.

Se invece la compilazione ha successo, l'OpenPLC Editor inizierà a eseguire il tuo codice.

Per visualizzare l'esecuzione del programma in modo interattivo, clicca su **Debug instance** nel pannello di sinistra:



Questo aprirà una **nuova finestra** in cui è possibile vedere il **flusso elettrico** del programma.

Le linee in **verde** sono **attivate**, mentre quelle in **nero** non lo sono.

E' possibile **forzare** l'attivazione o la disattivazione di un contatto o di una bobina cliccandoci sopra con il tasto destro e selezionando **Force True** oppure **Force False**.

Prova a forzare **PB1** su *True* e osserva il percorso del flusso fino alla bobina **LAMP**.

Poi forza di nuovo **PB1** su *False* e verifica che la **LAMP** rimanga comunque accesa grazie al **circuito di autoritenuta (latch circuit)**.

Oltre a visualizzare graficamente il flusso elettrico nel diagramma si può inoltre **monitorare i dati delle variabili** del programma nel **pannello Debugger** situato sul lato destro dello schermo. Si possono aggiungere variabili al pannello Debugger cliccando sull'**icona degli occhiali** davanti

a ciascuna variabile nel pannello di sinistra.

Inoltre, facendo **doppio clic su una variabile** nel pannello Debugger, viene mostrato un **grafico in tempo reale** che mostra il valore corrente della variabile.

Questa funzione è particolarmente utile quando il programma **conteggia passi o manipola dati**.

Infine, dopo aver **creato e testato** il programma, l'ultimo passaggio è **generarlo in un formato compatibile con l'OpenPLC Runtime**.

Per farlo basta cliccare semplicemente su **Generate program for OpenPLC Runtime** nella barra degli strumenti (icona freccia arancione verso il basso) e salvare il file **.st** sul computer.

Questo file rappresenta il programma in **logica ladder** scritto in un linguaggio che l'**OpenPLC Runtime** è in grado di comprendere.

Successivamente sarà possibile **caricare questo file** nell'**OpenPLC Runtime** seguendo le istruzioni indicate in **4.2 Caricare programmi in Open PLC Runtime**

Per le piattaforme più semplici, come le **schede Arduino**, è possibile caricare il programma direttamente da **OpenPLC Editor** cliccando sull'icona di **Arduino** nella barra degli strumenti superiore.

## 4. OPEN PLC RUNTIME - GUIDA AL PROGRAMMA

---

[Open PLC - caricare programmi su Open PLC Runtime](#)

### 4.1 Primo avvio

---

[Open PLC Runtime Overview](#)

L'**OpenPLC Runtime** permette di eseguire programmi PLC creati con l'**OpenPLC Editor**. Il **runtime principale** include un **web server integrato** che consente di configurare diversi parametri del runtime.

Le **implementazioni "micro"** dell'**OpenPLC Runtime** ( ovvero le versioni del runtime pensate per **microcontrollori e schede Arduino** che vengono eseguite sugli stessi) [non dispongono del web server integrato](#) poichè versioni ridotte dello stesso software per favorirne la portabilità.

In questi casi, tutte le **configurazioni del runtime micro** vengono effettuate **direttamente dalla finestra di caricamento (upload dialog)** dell'**OpenPLC Editor** (vedi [\\*\\*1.5 Installing OpenPLC Runtime on Microcontroller Boards\\*\\*](#)).

#### 4.1.1 - Primo accesso all'interfaccia

Il web server del runtime principale è accessibile all'indirizzo IP del dispositivo di destinazione sulla porta 8080.

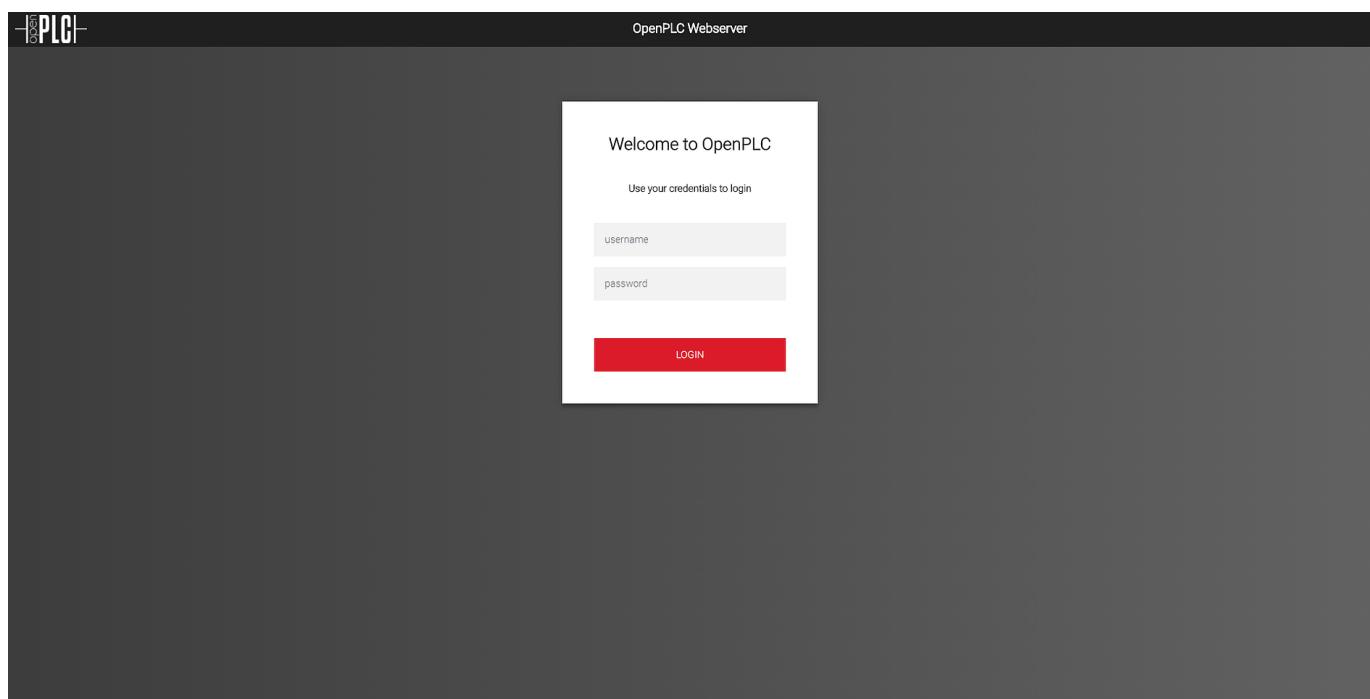
Ad esempio, se l'OpenPLC Runtime è stato installato su un **Raspberry Pi** e l'indirizzo IP del Raspberry è **192.168.0.103**, è possibile accedere al runtime aprendo il browser e visitando:  
<http://192.168.0.103:8080>



Se vengono visualizzati **errori di pagina**, è necessario verificare che il **computer abbia accesso al Raspberry Pi** all'interno della rete.

Nel caso in cui l'indirizzo IP della scheda non sia noto, la **Raspberry Pi Foundation** mette a disposizione [una guida utile](#) per individuarlo.

Una volta effettuato l'accesso al **web server di OpenPLC**, nel browser dovrebbe comparire una **pagina di login** simile alla seguente:



Il nome utente e la password predefiniti sono entrambi **openplc**. Questo implica che la **prima operazione da effettuare al primo accesso è cambiare il nome utente e la password predefiniti**.

La procedura è molto semplice: basta accedere al **menu Users** sulla sinistra e cliccare sull'**utente OpenPLC** per modificare le informazioni dell'utente secondo le necessità.

Una volta salvate le modifiche, verrà richiesto di effettuare nuovamente il **login**. Basterà inserire le **nuove credenziali** per poter accedere correttamente al sistema.

#### 4.1.2 - Abilitazione dell'accesso all'Hardware IO

Di default, OpenPLC Runtime viene installato con un **driver vuoto**.

Questo significa che, inizialmente, **non sarà possibile controllare direttamente i pin GPIO dell'hardware** tramite OpenPLC.

È necessario **abilitare il driver hardware corretto** per la piattaforma in uso. Nel menu a sinistra, cliccare su “**Hardware**” e selezionare il driver appropriato dal menu a comparsa. È importante scegliere il **driver corretto per la scheda**, altrimenti la compilazione del **runtime core** di OpenPLC non andrà a buon fine.

```

// KNOW WHAT YOU'RE DOING, JUST DON'T DO IT, EDIT AT YOUR OWN RISK.
// PS: You can always restore original functionality if you broke something
// in here by clicking on the "Restore Original Code" button above.
// -----
// These are the ignored I/O vectors. If you want to override how OpenPLC
// handles a particular input or output, you must put them in the ignored
// vectors. For example, if you want to override %IX0.5, %IX0.6 and %IM3
// your vectors must be:
// int ignored_bool_inputs[] = {5, 6}; //%IX0.5 and %IX0.6 ignored
// int ignored_int_inputs[] = {3}; //%IM3 ignored
// -----
// Every I/O on the ignored vectors will be skipped by OpenPLC hardware layer
// -----
int ignored_bool_inputs[] = {-1};
int ignored_bool_outputs[] = {-1};
int ignored_int_inputs[] = {-1};
int ignored_int_outputs[] = {-1};

```

Una volta selezionato il **driver corretto**, cliccare su “**Save changes**” e attendere che il **runtime core** venga ricompilato. Se tutto è stato configurato correttamente, alla fine dovrebbe comparire un **messaggio** che indica che la **compilazione è terminata con successo**.

Di seguito viene presentata una tabella riassuntiva dei driver resi disponibili da Open PLC e i relativi hardware supportati:

Driver OpenPLC	Hardware consigliato / piattaforma
Blank	Nessuno specifico, driver vuoto, utile per test senza hardware
Blank with DNP3 (Linux only)	Sistemi Linux generici che richiedono supporto DNP3
Fischertechnik	Piattaforme Fischertechnik PLC
PiXtend	Schede PiXtend per Raspberry Pi
PiXtend 2s	Schede PiXtend 2s per Raspberry Pi
Raspberry Pi	Raspberry Pi (modelli moderni, es. 3B, 4B)
Raspberry Pi - Old Model (2011 B)	Raspberry Pi modello B del 2011
Simulink	Ambienti di simulazione Simulink
Simulink with DNP3 (Linux only)	Simulink su Linux con supporto DNP3
UniPi v1.1	Schede UniPi v1.1

## 4.2 Driver e gestione degli I/O in OpenPLC Runtime

In **OpenPLC Runtime**, i **driver hardware** consentono al sistema di comunicare con dispositivi fisici esterni, come schede di acquisizione, microcontrollori o PLC industriali.

Tuttavia, i **driver non sono sempre necessari**, poiché OpenPLC integra un **livello hardware nativo** in grado di gestire autonomamente molte schede già supportate dal sistema.

OpenPLC traduce internamente le variabili logiche definite nel programma PLC (come `%IX0.0` , `%QX0.3` , `%IW2` , `%QW4` , ecc.) nei corrispondenti **pin fisici** delle schede compatibili.

Questa traduzione è gestita attraverso un meccanismo chiamato **Physical Addressing**, che stabilisce la corrispondenza tra la notazione PLC standard e le **uscite o ingressi reali** — digitali o analogici — presenti sull’hardware.

Il processo avviene automaticamente durante il **ciclo di scansione**, consentendo al programma di operare con indirizzi logici indipendenti dall'hardware utilizzato.

Nella configurazione di default, OpenPLC aggiorna in modo sincrono le **immagini di processo** degli ingressi e delle uscite (PII e PIQ), sincronizzandole con i moduli fisici a ogni ciclo.

Questo meccanismo garantisce coerenza dei dati e stabilità operativa, senza richiedere interventi manuali.

L'uso di **driver aggiuntivi** diventa necessario **solo quando OpenPLC deve comunicare con PLC esterni**, sistemi operativi non nativamente supportati o dispositivi personalizzati non inclusi nella lista ufficiale.

In questi casi, il driver funge da *ponte di comunicazione* tra OpenPLC Runtime e il protocollo o l'hardware specifico (ad esempio **Modbus TCP**, **EtherCAT**, o **OPC UA**).

OpenPLC supporta **nativamente** numerose piattaforme, tra cui:

- **Arduino UNO / Mega / Leonardo**
- **Raspberry Pi (tutte le versioni)**
- **ESP32 / ESP8266**
- **Controllino / UniPi / RevPi**
- **Schede Modbus RTU/TCP standard**

Per ciascuna di queste schede, OpenPLC fornisce una **tabella di conversione ufficiale** che mostra come gli indirizzi PLC vengono associati ai pin fisici di ingresso e uscita (digitali o analogici).

Queste tabelle sono consultabili direttamente nella documentazione ufficiale alla sezione *Physical Addressing*.

---

🔗 Consulta le tabelle di conversione ufficiali qui:

👉 <https://autonomylogic.com/docs/2-4-physical-addressing/>

---

---

💡 **Modbus: cos'è e a cosa serve**

Modbus è un protocollo di comunicazione seriale molto utilizzato in automazione industriale per lo scambio di dati tra PLC, sensori, attuatori e HMI.

Come funziona:

- Modello **master/slave** (o client/server): il master invia richieste, gli slave rispondono.

- I dati sono organizzati in **registri** e **coil**, che rappresentano ingressi/uscite digitali o valori analogici.
- Varianti principali:
  - **Modbus RTU**: su linee seriali RS-232 o RS-485.
  - **Modbus TCP**: su rete Ethernet.

A cosa serve:

- Far dialogare dispositivi di marche diverse senza software proprietario.
  - Monitoraggio e controllo remoto di sistemi industriali.
  - Simulare ingressi/uscite PLC in software come Node-RED senza dispositivi fisici.
-