DOMOTICA LABS

IKON SERVER

MODULO LOGICO AVANZATO

MANUALE DI CONFIGURAZIONE

VERSIONE SOFTWARE 2.9.0

INDICE

1 INTRODUZIONE....6

1.1 Convenzioni....6

1.2 Informazioni sulla sicurezza....6

1.3 Copyright....6

2 PANORAMICA GENERALE....7

2.1 Premessa....7

2.2 PROGRAMMI E TASK....7

2.3 MODALITÀ DI ESECUZIONE....8

3 INTERFACCIA UTENTE....9

3.1 Layout generale....9

3.2 Menu principale....9
3.2.1 Modulo logico avanzato....9
3.2.2 Libreria....10
3.2.3 Strumenti di disegno....10

3.3 Toolbar....10

3.4 Pannello dettagli....11

3.5 Area di lavoro....11

3.6 Area messaggi....12

4 PROGRAMMI LOGICI....14

4.1 Introduzione....14

4.2 Creazione di un nuovo programma logico....14

4.3 Creazione di un nuovo task....15

4.4 Rimozione o disattivazione di un programma....15

4.5 Aggiunta di blocchi ad un TASK....15

4.6 Selezione di uno o più blocchi....16

4.7 Rimozione di uno o più blocchi....17

4.8 Nodi di ingresso e uscita....18
4.8.1 Blocchi logici....18

4.9 Collegamento dei blocchi....19

4.10 Ordine di esecuzione....20

4.10.1 Ordine dei programmi....21
4.10.2 Ordine dei blocchi....21

4.11 Passaggio di valori tra programmi....22

4.12 Tipi di dato....23

5 LOGICHE....24

5.1 Introduzione....24

5.2 Blocchi logici....24
5.2.1 Layout....24
5.2.2 Nodi di ingresso....24
5.2.3 Nodi di uscita....24
5.2.4 Aggiunta e rimozione nodi....25

5.3 Logiche combinatorie....25
5.3.1 AND....25
5.3.2 OR....25
5.3.3 XOR....26
5.3.4 NOT....26

5.4 Scenari e sequenze....27
5.4.1 Sequencer....27
5.4.2 Scenario binario....27
5.4.3 Scenario numerico....28

5.5 Gates....29
5.5.1 Selettore binario....29
5.5.2 Selettore numerico....30
5.5.3 Encoder binario....30
5.5.4 Encoder numerico....31
5.5.5 Decoder (o selettore numerico)....31
5.5.6 Flip-flop T....32
5.5.7 Flip-flop RS....33
5.5.8 Flip-flop D....33
5.5.9 Latch D binario....34
5.5.10 Latch D numerico....34

5.6 Confronti....35
5.6.1 Operatori di confronto....35

5.7 Operazioni....36
5.7.1 Operatori matematici....36
5.7.2 Range....36

5.8 Contatori....37
5.8.1 Contatore su, contatore giù, contatore su/giù....37

5.9 Timers....38
5.9.1 Timer....38
5.9.2 Trigger....39
5.9.3 Trigger numerico....39
5.9.4 Orologio astronomico....39
5.9.5 Data/ora....40
5.9.6 Ripetizione comando....41
5.9.7 Oscillatore....41

5.10 Variabili....42
5.10.1 Premessa....42
5.10.2 Variabili binarie....42
5.10.3 Variabili numeriche....42

5.11 Costanti....42
5.11.1 Premessa....42
5.11.2 Costanti....43

6 TRIGGER....44

6.1 Introduzione....44

6.2 Trigger di ingresso ed uscita....44

6.3 Strategia di output....45

6.4 Connessioni trigger....45

6.5 Trigger e compositi....46

7 CONNESSIONI....48

7.1 Introduzione....48

7.2 Tipi di connessione....48

7.3 Condizioni sulle connessioni....49

7.4 Connessioni e codice LUA....50

8 SIMULAZIONE ED ESECUZIONE....51

8.1 Introduzione....51

8.2 Tipi di simulazione....51

8.3 Ambiente grafico di simulazione....51

8.4 Inserimento manuale dei valori....52

8.5 Arresto della simulazione....52

8.6 Esecuzione....52

9 STRUMENTI DI DISEGNO....54

9.1 Introduzione....54

9.2 Etichette....54

9.3 Aree rettangolari....55

10 CODICE LUA....56

10.1 Premessa....56

10.2 Struttura dello script....56

10.3 Lettura ingressi.....57

10.4 Connessioni.....58

10.5 Blocchi logici.....59

10.6 Output e debug.....61

11 APPENDICI.....63

11.1 Glossario.....63

1 INTRODUZIONE

1.1 CONVENZIONI

In questo documento sono adottate le seguenti convenzioni:

[KEY]	I tasti della tastiera che l'utente deve premere sono riportati tra parentesi quadra (es: [CTRL])
`Courier`	Messaggi a video o altri messaggi software sono riportati con il font “courier”
“...”	Nomi di pulsanti o altri elementi dell'interfaccia grafica sono riportati tra doppio apice
	Nota particolarmente importante

1.2 INFORMAZIONI SULLA SICUREZZA

Questo manuale contiene le informazioni necessarie per poter operare in sicurezza con il dispositivo. Chiunque interagisca con il dispositivo deve prima aver letto questa documentazione, in particolar modo le presenti informazioni sulla sicurezza. Questo documento integra e non sostituisce qualunque normativa o direttiva di Legge in materia di sicurezza.

Il dispositivo è stato sviluppato utilizzando tecnologie allo stato dell'arte e seguendo le normative in termini di sicurezza vigenti al momento; non è possibile tuttavia escludere con certezza possibili danni o interazioni con altri dispositivi durante il suo esercizio. Il dispositivo è conforme alle guide linea EMC ed alle norme Europee armonizzate; eventuali modifiche al dispositivo possono inficiare la compatibilità EMC.

La tensione di alimentazione deve essere rigorosamente all'interno del range indicato in questo manuale e sul dispositivo; pericolo di incendio o di esplosione in caso di alimentazione al di fuori di questo range. L'apparecchiatura deve essere alimentata da un circuito di alimentazione limitato il cui isolamento dalla rete elettrica deve essere non inferiore a quello tra il circuito primario e secondario di un trasformatore di sicurezza secondo IEC 61558-2-6 o equivalente.

La dichiarazione di conformità CE del dispositivo può essere richiesta a Domotica Labs, ai riferimenti riportati sul sito www.domoticalabs.com



In conformità con la direttiva 2002/96/EC i dispositivi elettronici devono essere smaltiti nelle apposite strutture e non nella raccolta di rifiuti solidi urbani.

1.3 COPYRIGHT

Le istruzioni tecniche, i manuali ed il software di IKON SERVER sono soggetti a copyright; tutti i diritti sono riservati. La copia, riproduzione, traduzione e/o modifica, anche parziali, sono espressamente vietati salvo approvazione scritta da parte di Domotica Labs.

2 PANORAMICA GENERALE

2.1 PREMESSA

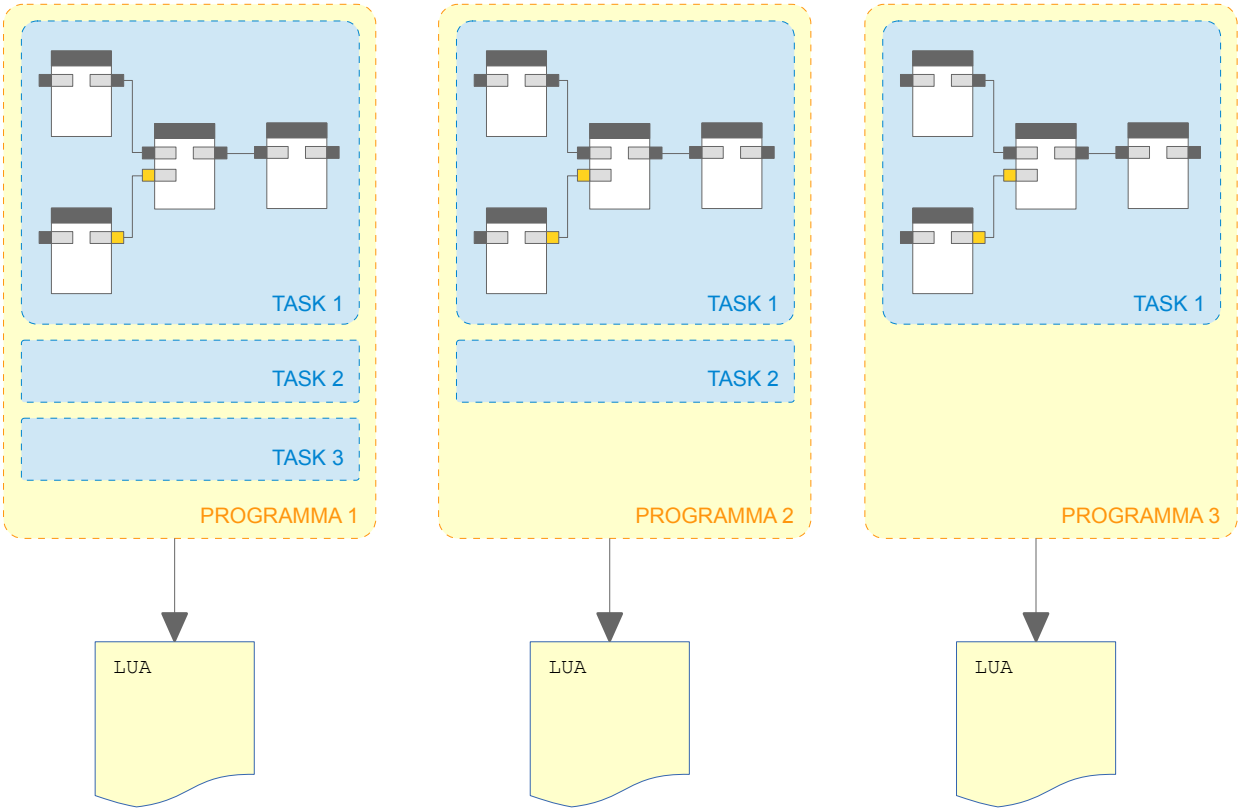
Il MODULO LOGICO AVANZATO permette di realizzare, all'interno di IKON SERVER, reti logiche complesse tramite un editor grafico integrato ed una ricca libreria di blocchi logici, con i quali interconnettere oggetti di qualunque tecnologia tra quelle gestite dal webserver.

2.2 PROGRAMMI E TASK

Il modulo permette di creare uno o più **programmi logici**, i quali a loro volta possono essere costituiti da uno o più **task**; ogni task è una rete logica configurabile attraverso un editor grafico, simile agli *eventi programmabili* di IKON SERVER, che può scambiare dati con gli altri task attraverso appositi oggetti di tipo variabile, come meglio specificato in seguito.

I task possono essere abilitati o meno, e in base a tale abilitazione, quando si lancia (in simulazione oppure in esecuzione "reale") un programma, viene generato un listato in linguaggio LUA, il quale viene messo in esecuzione in background, e continua a ciclare scambiando dati in ingresso / uscita, in modo concettualmente simile a quanto avviene nei PLC.

I programmi logici possono essere messi in esecuzione in parallelo, essendo a tutti gli effetti processi indipendenti; i task viceversa, sono "porzioni" di un programma logico, e confluiscono nel listato LUA in fase di compilazione, e pertanto vengono sempre eseguiti tutti, dal primo all'ultimo, se abilitati, in base all'ordine stabilito nella pagina di configurazione del programma.



2.3 MODALITÀ DI ESECUZIONE

A differenza delle altre funzionalità logiche di IKON SERVER, che vengono processate “ad evento” (ovvero nel momento in cui si genera un cambio di stato su uno degli ingressi), **i programmi logici LUA sono perennemente in esecuzione.**

Il codice LUA viene seguito ciclicamente in background, e ad ogni ciclo di esecuzione, elabora la logica in base allo stato degli oggetti in ingresso rilevato all'inizio del ciclo; gli oggetti in uscita vengono comandati di conseguenza, se il loro stato non è già coerente con quanto previsto dalla logica.

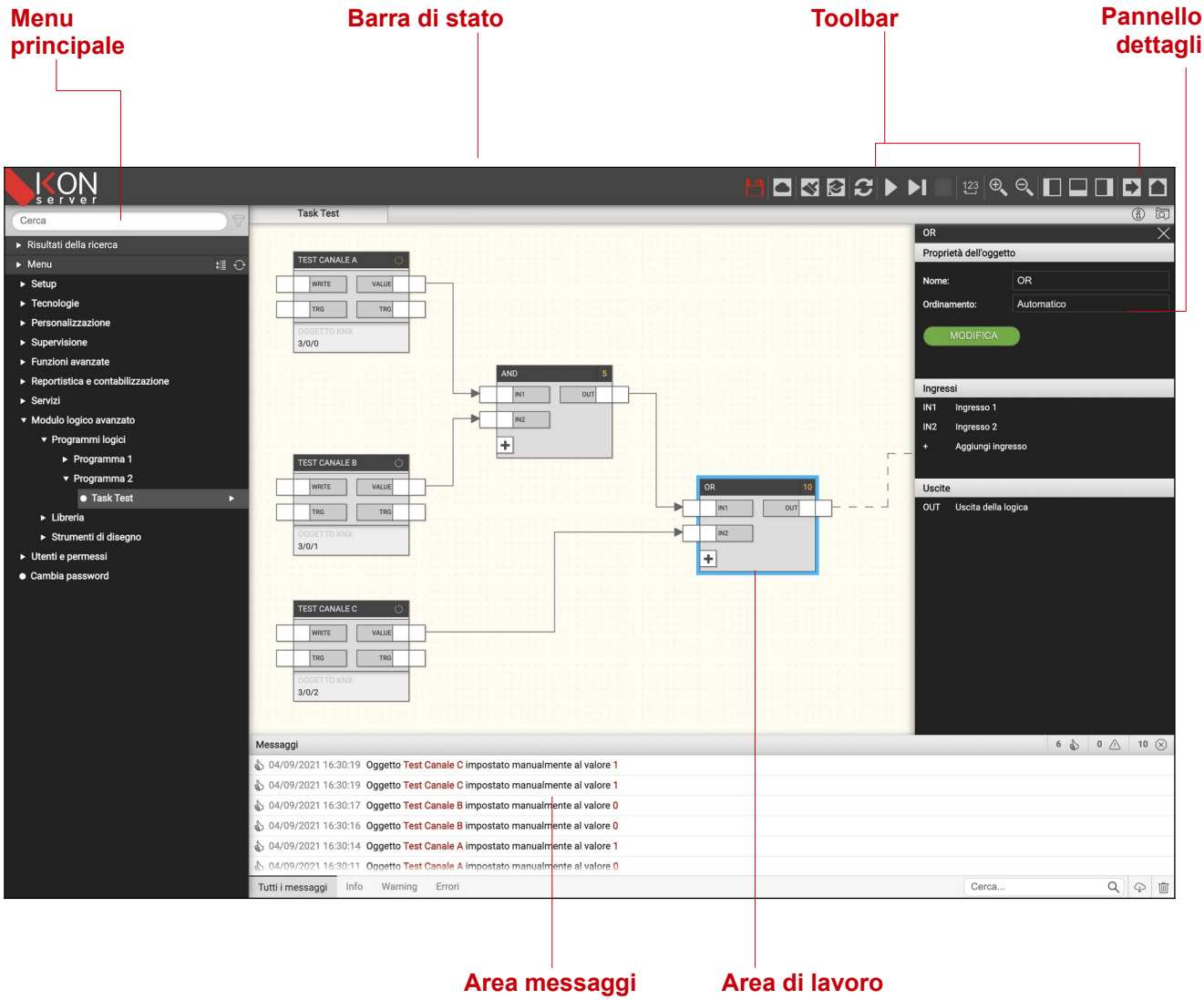
In condizioni normali, quindi, gli oggetti in uscita vengono quindi **mantenuti allineati con la logica** anche se il loro stato varia per ragioni esterne, mandando immediatamente un comando (al ciclo di esecuzione LUA successivo); questo comportamento è molto diverso dalle altre logiche ad evento di IKON SERVER, e deve essere tenuto presente quando si crea una rete logica all'interno del modulo logico avanzato.

E' possibile tuttavia regolamentare quando la logica LUA debba comandare gli oggetti di uscita, ad esempio solo se variano gli oggetti di ingresso (evitando quindi di “bloccare” le uscite se queste variano per ragioni differenti, ad esempio perché comandate dall'utente), utilizzando i cosiddetti TRIGGER; un apposito capitolo, nell'ultima parte di questo manuale, spiega come utilizzarli; si consiglia una lettura approfondita di questa sezione.

3 INTERFACCIA UTENTE

3.1 LAYOUT GENERALE

La figura seguente mostra la struttura dell'interfaccia grafica dell'editor, una volta aperta la sua finestra:



3.2 MENU PRINCIPALE

Il menu mette a disposizione tutti gli strumenti per la creazione e gestione dei programmi logici. I “tab” nella porzione iniziale del menu permettono di accedere alle sezioni principali del menu.

3.2.1 MODULO LOGICO AVANZATO

Questa sezione contiene la lista dei programmi logici configurati. Questa sezione del menu permette di creare, modificare ed eliminare i programmi logici.

3.2.2 LIBRERIA

Questa sezione contiene la libreria dei blocchi logici che è possibile inserire all'interno dei programmi. Come meglio descritto in seguito, le voci della libreria logica possono essere inserite nei programmi mediante drag&drop.

3.2.3 STRUMENTI DI DISEGNO

Questa sezione contiene gli strumenti con i quali possiamo personalizzare i nostri programmi. Come meglio descritto in seguito, gli strumenti di disegno possono essere inserite nei programmi mediante drag&drop.

3.3 TOOLBAR

La toolbar mette a disposizione i seguenti strumenti, sempre disponibili durante ogni fase della realizzazione dei programmi logici:



ESECUZIONE / PAUSA
Avvia o arresta un programma



SIMULAZIONE CONTINUA
Avvia la simulazione in modalità real-time



SIMULAZIONE STEP-BY-STEP
Avvia la simulazione in modalità passo-passo



ARRESTA SIMULAZIONE
Arresta la simulazione in corso



ORDINAMENTO
Riordina i blocchi mediante ordinamento automatico seguendo la posizione dei blocchi



ZOOM +
Ingrandisce il fattore di zoom dell'area di lavoro



ZOOM -
Riduce il fattore di zoom dell'area di lavoro



MOSTRA / NASCONDI MESSAGGI
Mostra o nasconde l'area messaggi in basso



MOSTRA / NASCONDI MENU PRINCIPALE
Mostra o nasconde il menu principale a sinistra



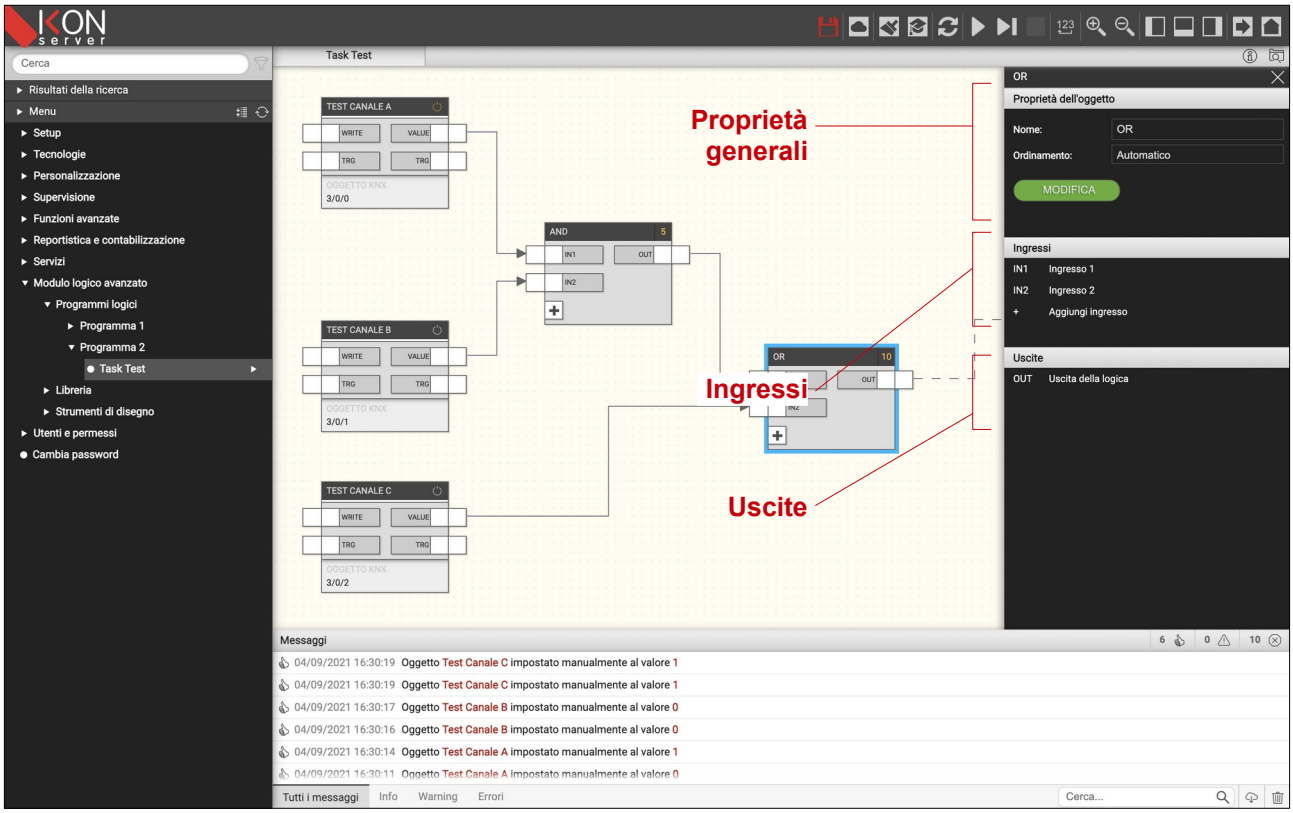
MOSTRA / NASCONDI DETTAGLI
Mostra o nasconde il pannello laterale contenente i dettagli

OPZIONI AVANZATE
Permette di accedere ad un menu a tendina contenente opzioni avanzate, come meglio dettagliate in seguito

3.4 PANNELLO DETTAGLI

Questa sezione, normalmente chiusa (può essere aperta tramite l'apposito pulsante nella toolbar), contiene i dettagli relativi agli oggetti selezionati nell'area di lavoro, e permette di modificarne proprietà ed opzioni.

A seconda del tipo di oggetto selezionato, le informazioni possono essere suddivise in più sezioni, come esemplificato nella figura seguente:



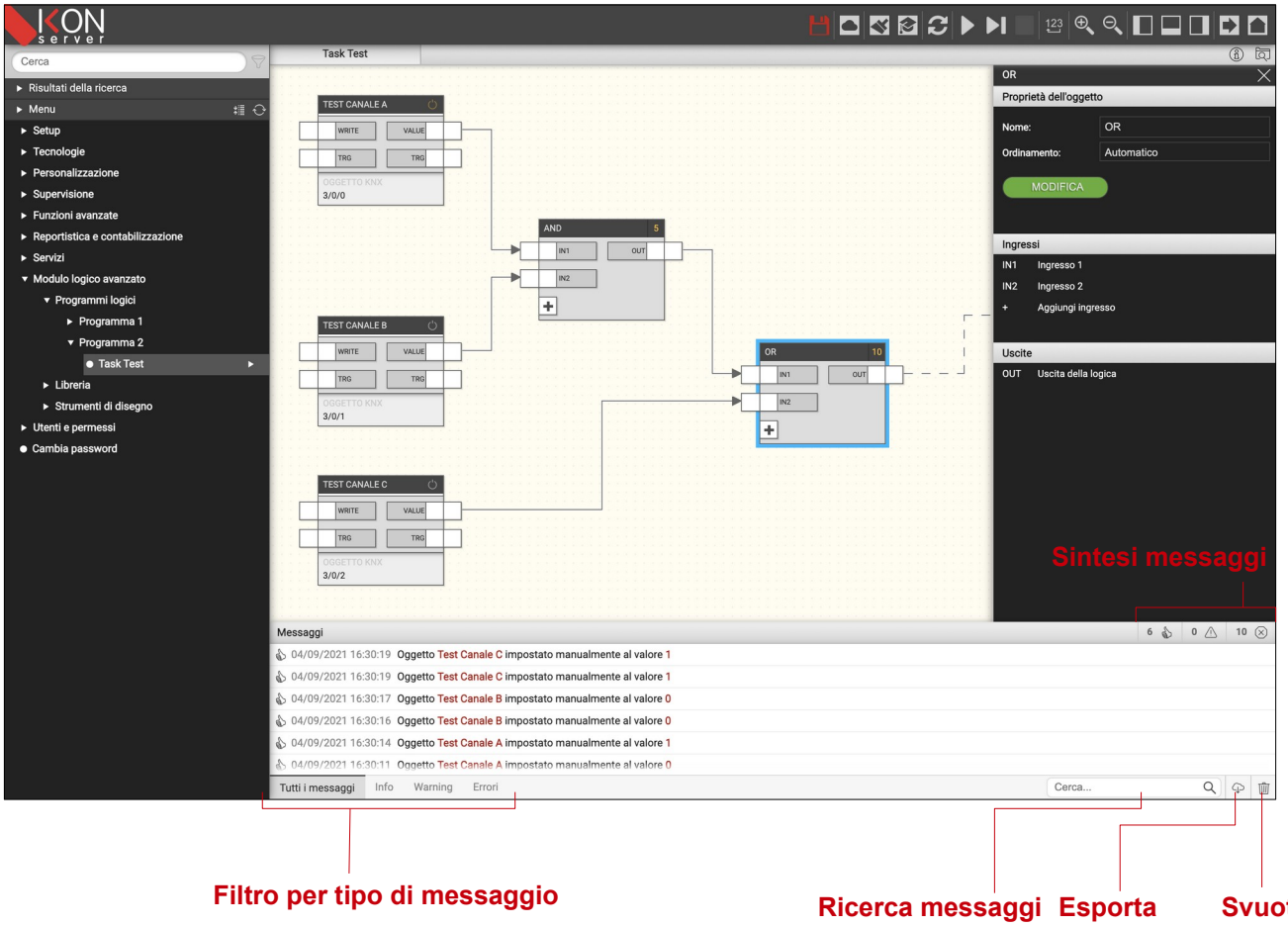
Le sezioni possono essere chiuse (facendo click sulla barra del titolo) per permettere una più agevole consultazione delle successive, soprattutto in caso di oggetti con molti dettagli ed opzioni.

3.5 AREA DI LAVORO

La parte centrale della finestra è dedicata allo spazio di lavoro, all'interno del quale costruire le logiche come meglio dettagliato in seguito. Per ampliare lo spazio utile, si consiglia di chiudere i pannelli laterali e l'area messaggi, soprattutto durante l' *editing* dei programmi logici.

3.6 AREA MESSAGGI

La parte inferiore della finestra ospita i messaggi generati dall'editor durante la realizzazione dei programmi logici e, soprattutto, durante la simulazione (come meglio dettagliato in seguito).



I messaggi generati dall'editor possono essere di diverso tipo, in base alla loro gravità e tipologia:

- **Error**: segnalazioni di operazioni o di condizioni che generano un errore, e che richiedono tipicamente una modifica o una verifica da parte dell'utente
- **Warning**: avvisi di condizioni anomale, le quali peraltro non necessariamente costituiscono un errore o una situazione da modificare
- **Info**: messaggi informativi “normali”, che segnalano operazioni effettuate dall'editor degne di segnalazione all'utente
- **Debug**: messaggi di dettaglio delle operazioni effettuate dalla simulazione (disponibili solo in modalità “step by step”, come meglio dettagliato in seguito)

Le diverse tipologie sono contraddistinte da un colore, evidenziato a lato di ogni messaggio insieme alla data/ora in cui il messaggio è stato generato. La barra del titolo dell'area messaggi contiene, nella parte destra, una sintesi del numero di messaggi delle diverse tipologie, visibile anche quando l'area messaggi è chiusa.

Nella parte inferiore dell'area messaggi sono disponibili i seguenti comandi:

- **Filtro per tipo di messaggio**: selezionando una delle voci disponibili è possibile filtrare i messaggi a video in base alla tipologia corrispondente
- **Ricerca messaggi**: permette di filtrare i messaggi in base ad una o più parole chiave
- **Esporta**: permette di esportare lo storico dei messaggi (anche quelli relativi a sessioni di lavoro precedenti) in formato CSV, consultabile tramite software esterni (es: fogli di calcolo)

- **Svuota**: permette di cancellare i messaggi a video (i messaggi rimangono comunque archiviati nell'editor e possono essere esportati tramite l'apposito pulsante per una consultazione “offline”)

4 PROGRAMMI LOGICI

4.1 INTRODUZIONE

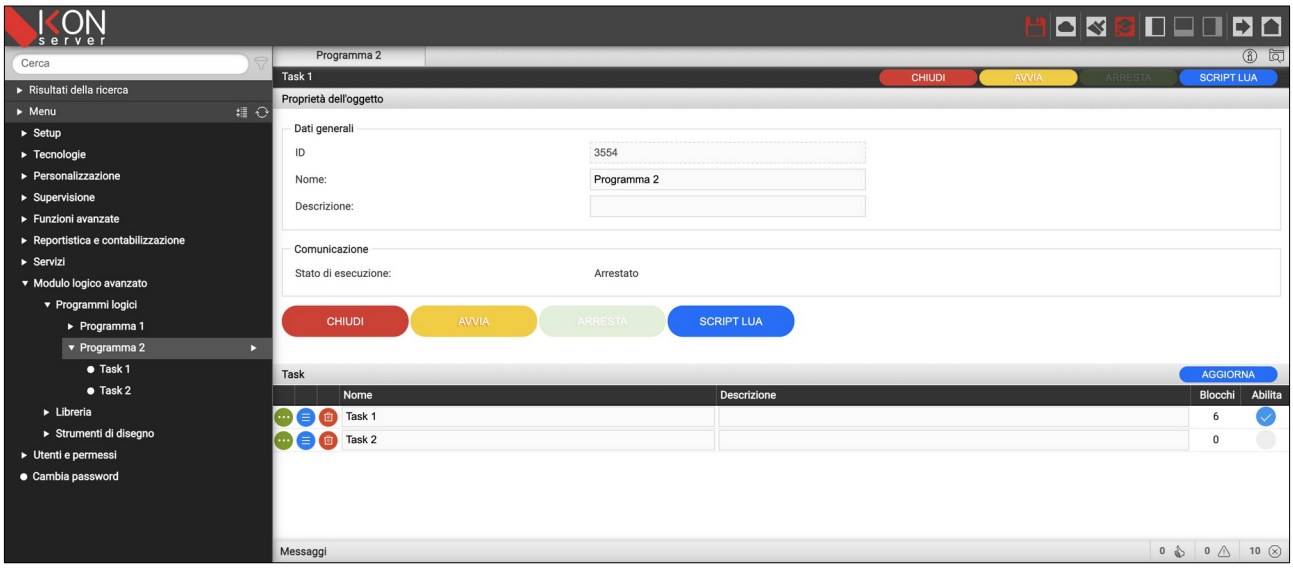
I moduli logici sono predisposti per eseguire una o più reti logiche, dette “task”, che tipicamente ricevono una o più informazioni dal bus, le elaborano attraverso blocchi logici, ed inviano i risultati sotto forma di comandi sul bus. Ogni programma può contenere fino ad un massimo di 16 task.

L'editor permette di configurare i programmi logici connettendo blocchi e funzioni logiche mediante *drag&drop* e semplici strumenti grafici, senza nessuna particolare nozione di programmazione. Come sarà illustrato più avanti, l'editor permette anche di simulare il comportamento dei programmi logici.

4.2 CREAZIONE DI UN NUOVO PROGRAMMA LOGICO

Per creare un nuovo programma logico, entrare nella sezione “Programmi logici” all’interno della sezione “Modulo logico avanzato” del menu principale, quindi premere il corrispondente pulsante “+”: viene creato un nuovo programma vuoto denominato “Programma 1”.

Aprendo la sua pagina di dettaglio, viene mostrata la seguente pagina:



La sezione principale della scheda permette di modificare il nome del programma, ed eventualmente impostare una descrizione (facoltativa); sono disponibili inoltre i seguenti pulsanti:

AVVIA ARRESTA	Permettono rispettivamente di avviare ed arrestare l’esecuzione del programma. Questa operazione può essere svolta anche all’interno dell’editor logico, come meglio illustrato in seguito.
SCRIPT LUA	Permette di visualizzare il codice generato dall’editor, nel pannello laterale; questo codice non può essere modificato manualmente, ma può essere utile per l’assistenza tecnica qualora il programma non svolga le funzionalità desiderate.

La sezione TASK nella porzione inferiore della pagina, viceversa, contiene il riepilogo dei task associati al programma, e permette di disabilitarli (vedere dopo). Viene inoltre indicato il numero di blocchi logici presenti all’interno.

4.3 CREAZIONE DI UN NUOVO TASK

Per creare un nuovo task, selezionare innanzitutto il programma logico su cui si vuole operare nella sezione “Programmi logici” del menu principale, quindi premere il corrispondente pulsante “+”: viene creato un nuovo programma vuoto denominato “Task 1”.

Per aprire il nuovo task è sufficiente fare click su di esso: nell'area di lavoro viene mostrata una griglia vuota, su cui è possibile iniziare la costruzione della logica, come descritto in seguito.

Per modificare il nome del task, aprire il pannello dei dettagli ed inserire il nuovo nome nell'apposita casella di testo, come evidenziato nella figura seguente; il nome non può contenere caratteri speciali e deve avere una lunghezza massima di 16 caratteri.



Se il programma risulta già aperto nell’area principale di lavoro e i nuovi task non compaiono nell’elenco, premere AGGIORNA per visualizzarli.

4.4 RIMOZIONE O DISATTIVAZIONE DI UN PROGRAMMA

Per rimuovere un programma esistente, è sufficiente premere il corrispondente pulsante “X” nel menu principale; una volta confermata la cancellazione, il programma viene eliminato, e con esso tutte le funzioni logiche presenti al suo interno. Questa operazione non può essere annullata.

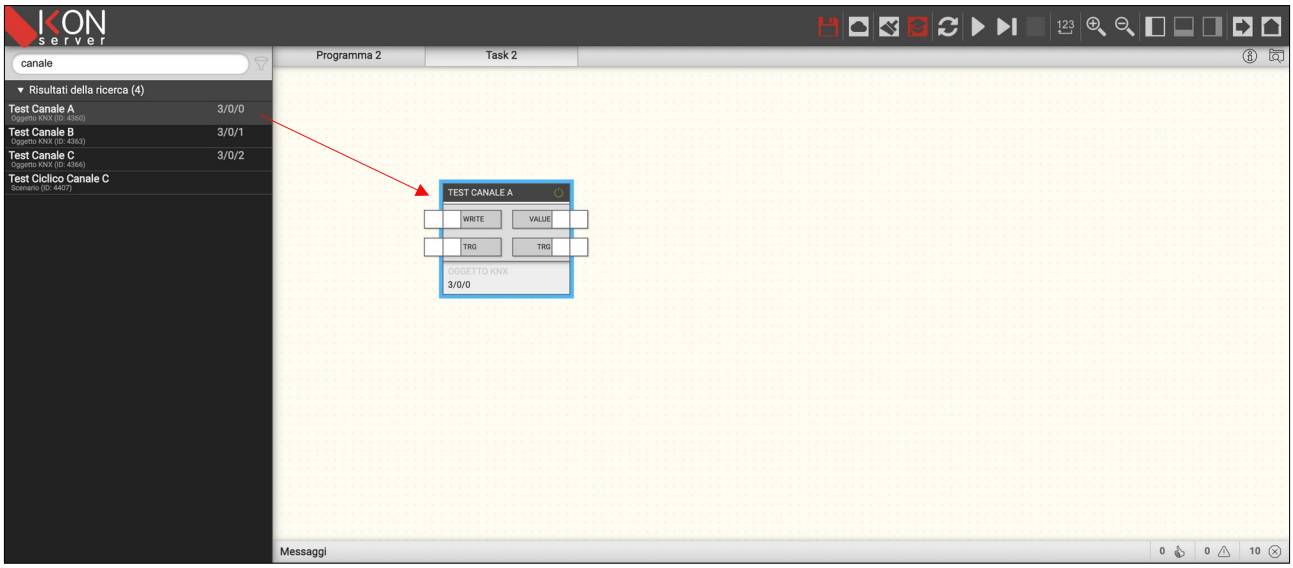
Se si desidera che un programma non venga inserito nell'unità logica, perché ad esempio ancora incompleto, è possibile disattivarlo deselezionando la voce “ABILITATO” corrispondente nella scheda del programma.

4.5 AGGIUNTA DI BLOCCHI AD UN TASK

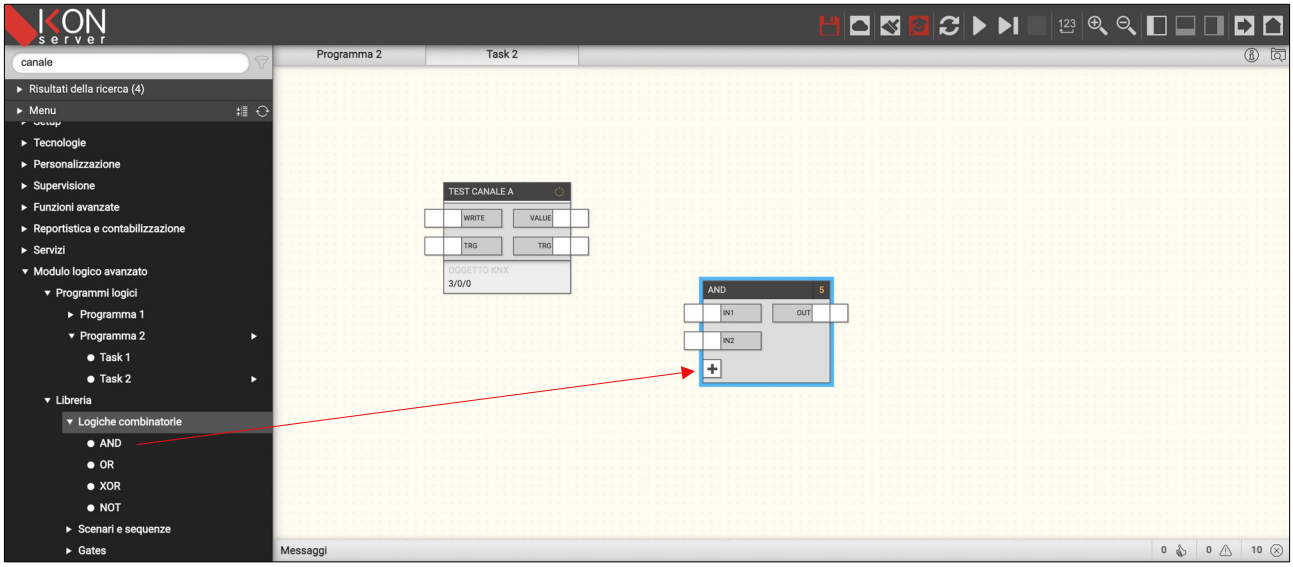
I programmi prevedono il collegamento di più blocchi a formare una rete logica. I blocchi possono essere di tipo logici; i primi sono necessari per leggere e/o scrivere informazioni sul bus domotico, i secondi permettono di elaborare e combinare queste informazioni.

Per aggiungere un blocco ad un task, è necessario innanzitutto identificarlo all'interno della sezione “Progetto ETS” che si trova nella sezione “KNX” interna a “Tecnologie” del menu principale; qui sono elencati tutti i gruppi supportati, suddivisi per tipologia.

Una volta identificato il blocco, è sufficiente trascinarlo nell'area di lavoro mediante *drag&drop*:



Per inserire un blocco logico, analogamente, è necessario innanzitutto identificarlo nella libreria “LOGICHE”, anch'essa organizzata per tipologia (per un elenco completo dei blocchi logici disponibili si rimanda al capitolo 5), per poi trascinarlo nello spazio di lavoro:

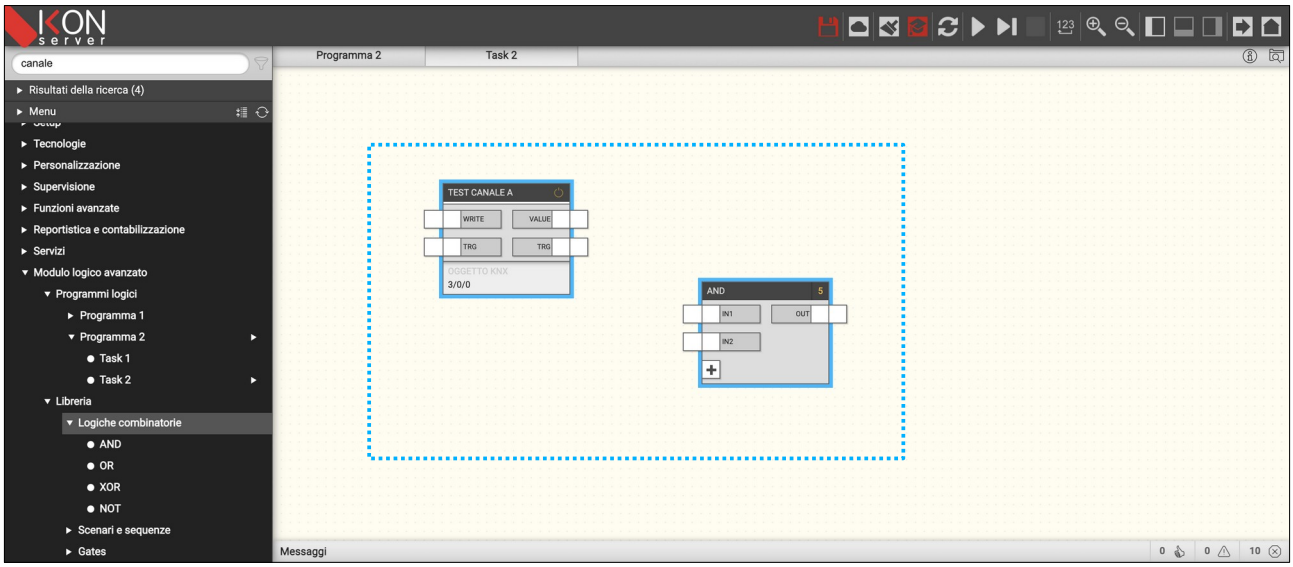


4.6 SELEZIONE DI UNO O PIÙ BLOCCHI

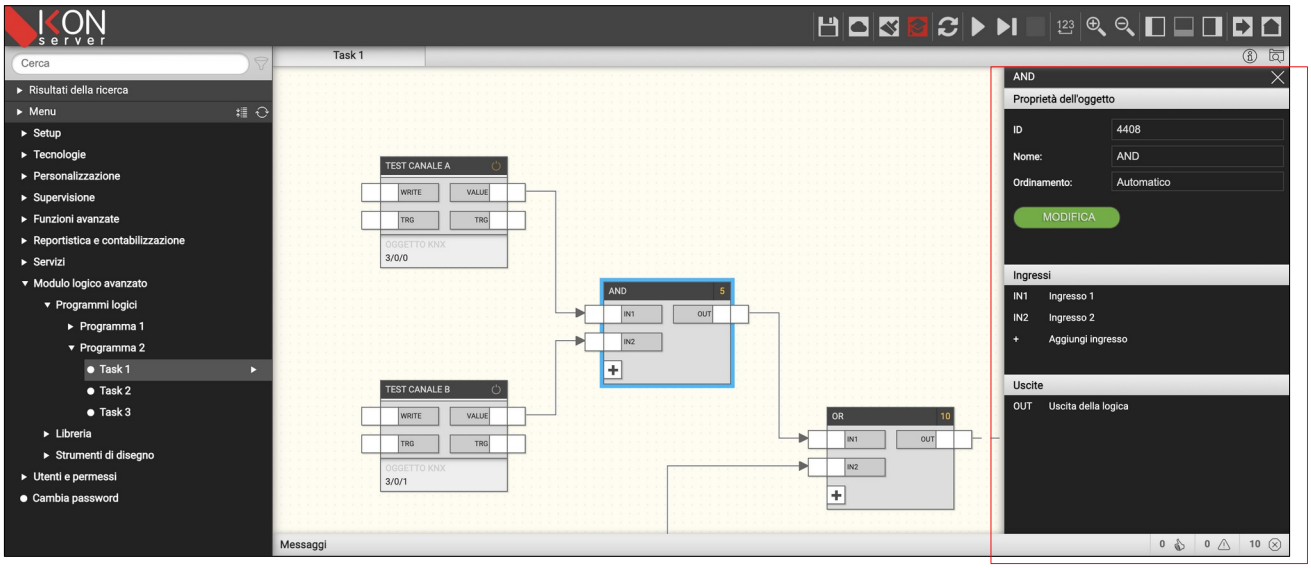
E' possibile selezionare uno o più blocchi presenti all'interno di un task, in diversi modi:

- Facendo click sul “titolo” del blocco (selezione singola)
- Facendo click sul “titolo” di più blocchi, tenendo contemporaneamente premuto il tasto CTRL (selezione multipla “sparsa”)
- Facendo click in un punto dell'area di lavoro e, tenendo premuto, spostare il cursore disegnando un'area di selezione rettangolare (selezione multipla “contigua”)

I blocchi selezionati vengono evidenziati con un bordo azzurro:



I blocchi selezionati possono essere spostati all'interno dell'area di lavoro semplicemente con il *drag&drop*. Selezionando un singolo blocco ed aprendo il pannello dei dettagli, viceversa, è possibile visualizzare le sue proprietà, l'elenco dei nodi di ingresso e di uscita, e gestire le eventuali opzioni:



NOTA: Selezionando più blocchi contemporaneamente non è possibile visualizzare i dettagli, essendo differenti per ognuno di essi.

4.7 RIMOZIONE DI UNO O PIÙ BLOCCHI

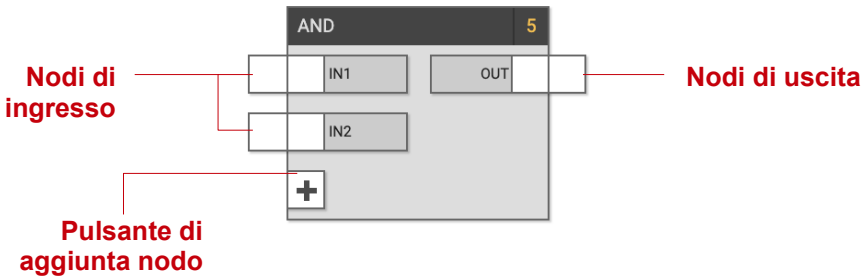
Per rimuovere uno o più blocchi da un programma, procedere in uno dei seguenti modi:

- Selezionare un singolo blocco, aprire il pannello dettagli e premere il pulsante “ELIMINA”
- Selezionare uno o più blocchi e premere il tasto “CANC” della tastiera

In entrambi i casi, dopo un messaggio di conferma, i blocchi selezionati vengono rimossi dal programma, così come le eventuali connessioni con altri blocchi presenti nel programma stesso. Questa operazione non può essere annullata.

4.8 NODI DI INGRESSO E USCITA

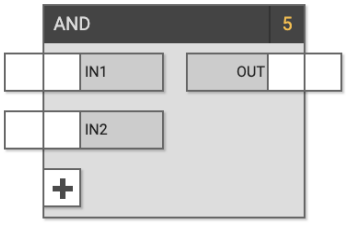
Ogni blocco contiene almeno un “nodo” di ingresso e/o uscita, come esemplificato nella figura seguente:



I nodi di ingresso sono sempre sul lato sinistro di un blocco, mentre le uscite sono a destra. Ogni nodo è caratterizzato da una etichetta sintetica (es: “IN1”, “IN2” e “OUT” nella figura precedente) che sono riportati nell'elenco ingressi / uscite nel pannello dei dettagli, assieme ad una sintetica descrizione di ogni nodo (oltre che all'interno di questo manuale).

4.8.1 BLOCCHI LOGICI

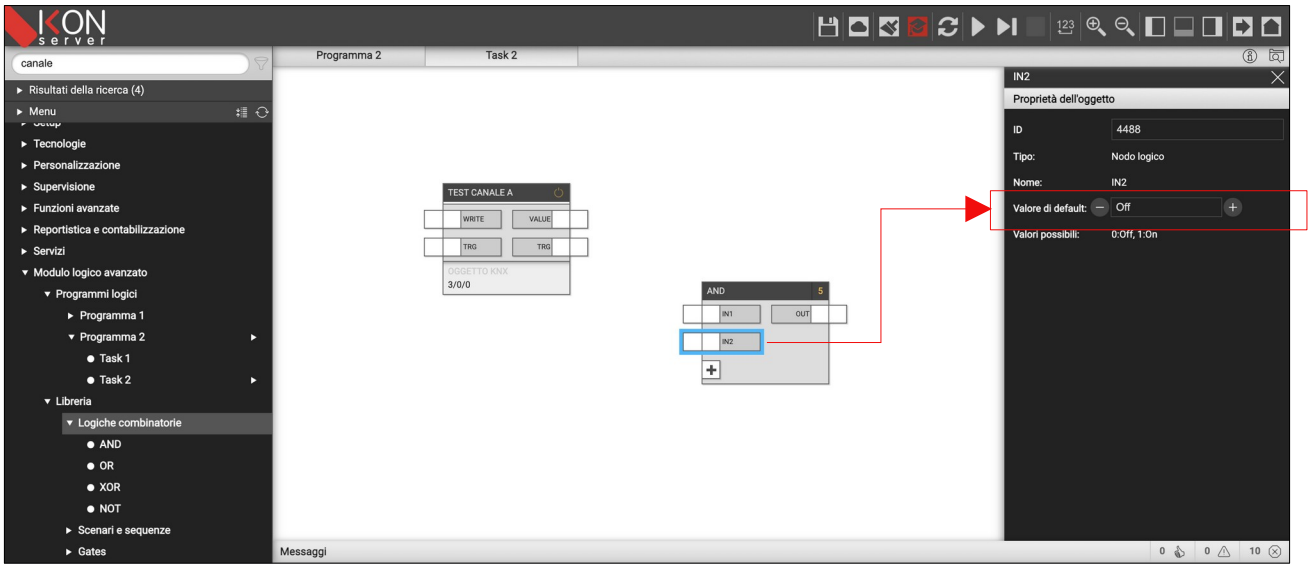
Nel caso dei blocchi logici, i nodi di ingresso rappresentano gli “input” alla funzione logica associata al blocco, mentre i nodi di uscita sono gli “output”:



In alcuni casi, come in questo esempio, il blocco prevede un numero variabile di nodi (di ingresso o di uscita); in questo caso, il pulsante “+” permette di aggiungere nodi al blocco, fino al numero massimo.

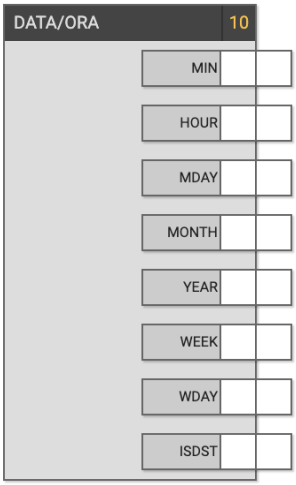
La funzione logica può essere eseguita correttamente solo se i nodi di ingresso sono collegati ad altri blocchi, e se i valori di uscita sono “riportati” sui nodi di ingresso di altrettanti blocchi.

Non tutti i nodi di ingresso sono strettamente necessari per la corretta esecuzione della logica; se un nodo di ingresso non è collegato, viene usato il suo valore di default, modificabile selezionando il nodo e aprendo il corrispondente pannello dettagli, come esemplificato nella figura seguente:



Il pannello di dettaglio di un nodo evidenzia inoltre i possibili valori che esso può assumere; questa informazione può essere utile soprattutto per blocchi che prevedono combinazioni o restrizioni specifiche di valori.

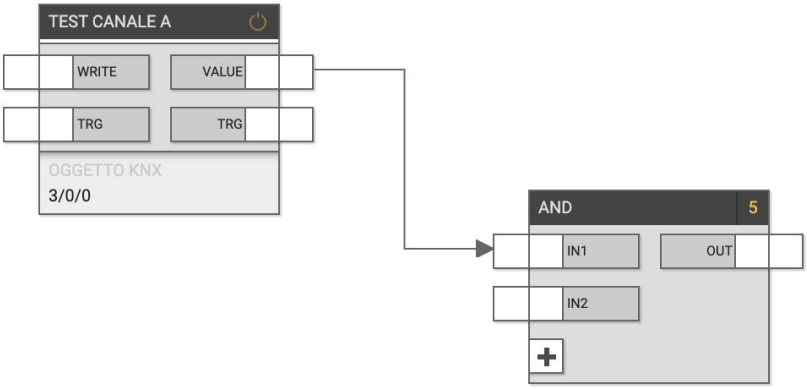
I blocchi logici possono anche prevedere solo uscite, come nell'esempio seguente (blocco di data/ora):



In questo caso, essi possono solo essere usati come ingresso per altre logiche, ma non possono essere comandati.

4.9 COLLEGAMENTO DEI BLOCCHI

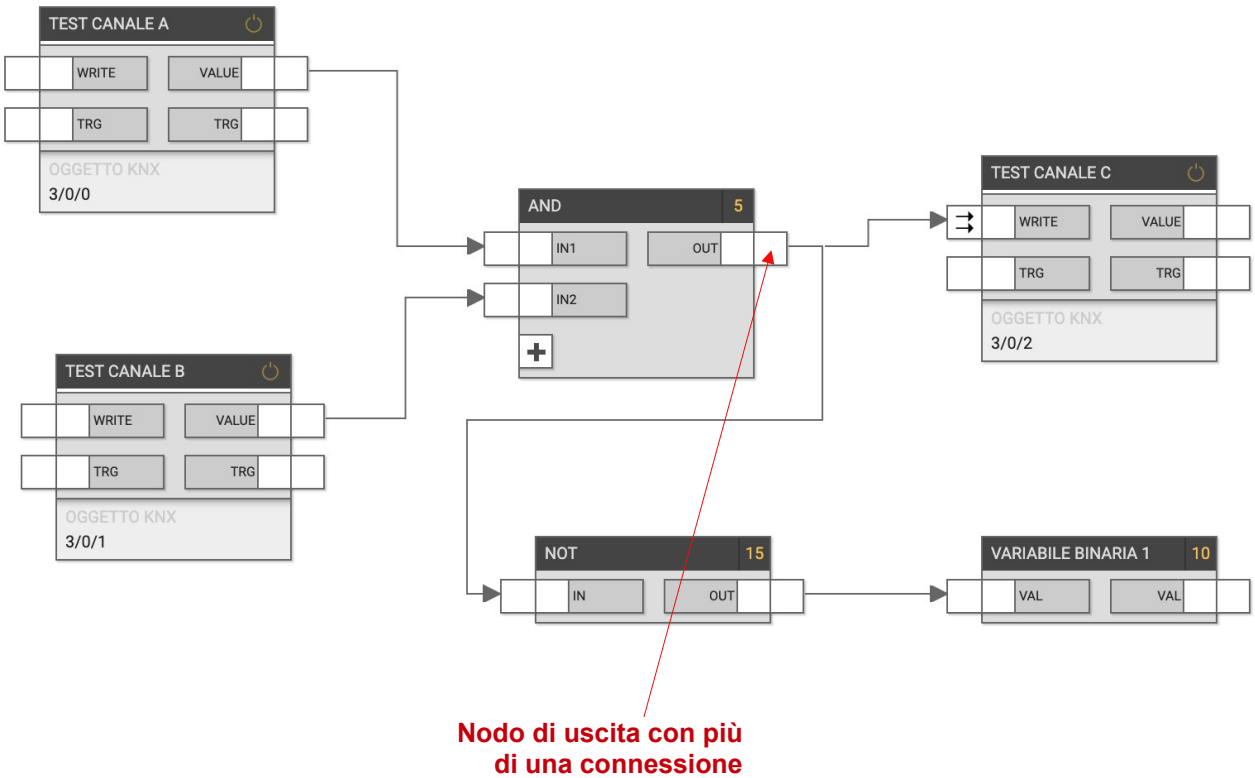
Affinché il programma svolga effettivamente qualche azione, è necessario prevedere almeno un “collegamento” tra due nodi di altrettanti blocchi, in modo che il valore del primo (“origine”) venga passato al secondo (“destinazione”). Per collegare due nodi, è sufficiente fare click al centro del nodo di origine, tenere premuto e rilasciare al centro del nodo di destinazione:



Passando con il cursore sopra un collegamento, questo viene evidenziato con il colore rosso (e passa “in primo piano” rispetto ad eventuali altre connessioni o blocchi lungo il suo percorso); facendo click su di esso, viceversa, il collegamento viene selezionato. Il collegamento selezionato può essere eliminato in due modi:

- Premendo il pulsante “ELIMINA” nel pannello dei dettagli
- Premendo direttamente il tasto “CANC” della tastiera

L'origine di un collegamento deve essere un nodo di uscita (lato destro di un blocco) mentre la destinazione deve essere un nodo di ingresso (lato sinistro); un nodo di uscita può essere origine di più collegamenti (con diverse destinazioni), mentre un nodo di ingresso può essere la destinazione di un solo collegamento:



4.10 ORDINE DI ESECUZIONE

Durante le fasi di simulazione e compilazione, come meglio dettagliato in seguito, l'editor genera, a partire dalle reti logiche disegnate graficamente, un “listato” che viene messo in esecuzione in

modo ciclico, dall'inizio alla fine, quanto più velocemente possibile (in base alla complessità del progetto).

4.10.1 ORDINE DEI PROGRAMMI

Ad ogni ciclo di esecuzione vengono effettuate le seguenti operazioni (il tempo di ciclo dipende dal numero e dalla complessità dei programmi):

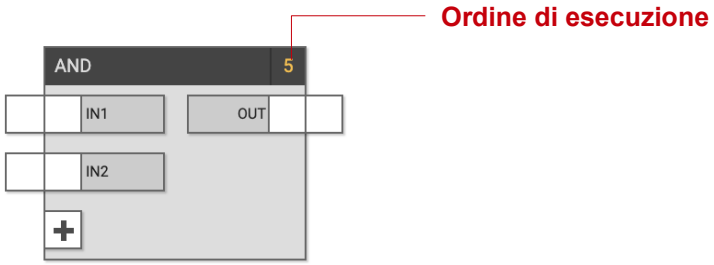
- Lettura degli ingressi dal bus
- Esecuzione del programma 1
- Esecuzione del programma 2
- …
- Esecuzione del programma n
- Scrittura dei comandi sul bus

L'ordine dei programmi è esattamente quello evidenziato nel menu principale; questo implica che eventuali interazioni tra i programmi (come ad esempio il passaggio di valori tramite variabili, oppure la scrittura dello stesso nodo di un blocco da parte di più programmi) sono influenzate da questo ordine (ed eventuali azioni effettuate dai programmi “in coda” all'elenco vengono recepite da quelli precedenti solo al successivo ciclo di esecuzione).

NOTA: Se un programma è disabilitato oppure è in pausa, esso viene “saltato” nel ciclo di esecuzione; ogni eventuale interazione con il bus e/o con altri programmi in questo caso è sospesa.

4.10.2 ORDINE DEI BLOCCHI

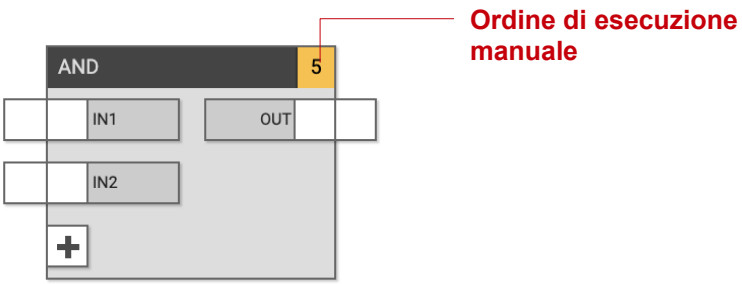
All'interno di ogni programma, anche i blocchi logici hanno un proprio ordine di esecuzione;il modulo logico elabora la funzione associata ai blocchi logici seguendo questo ordine. L'ordine di un blocco logico è evidenziato in alto a destra, come evidenziato nella figura seguente:



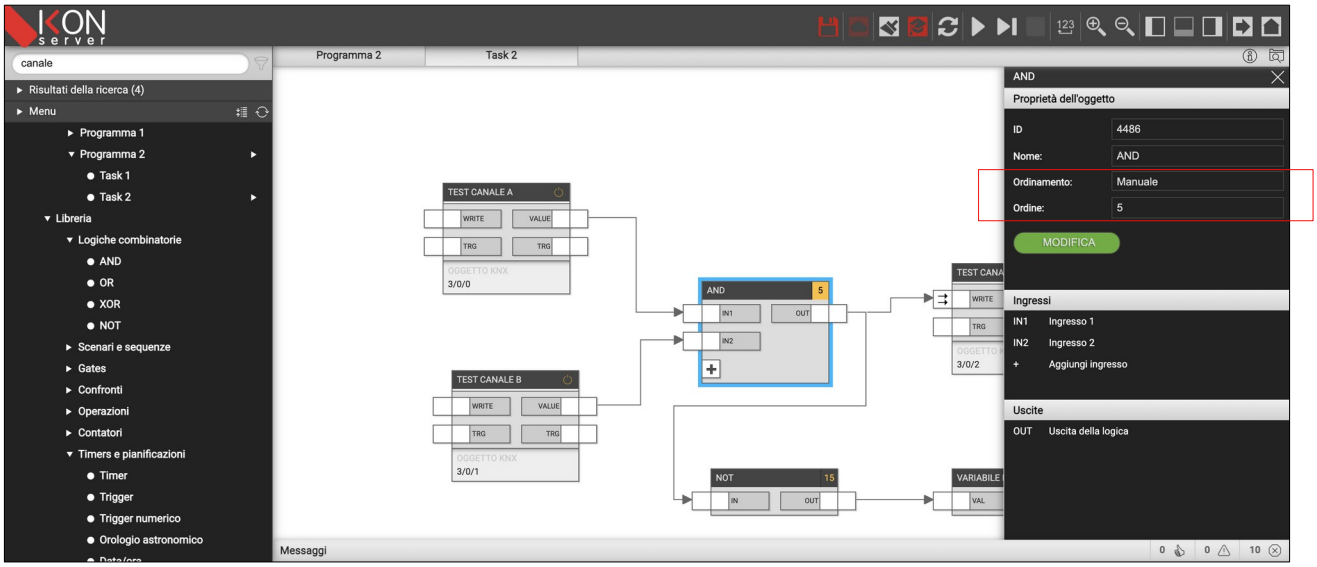
In condizioni normali, ai blocchi viene assegnato un ordine crescente in base all'ordine con cui vengono inseriti nel programma; è possibile tuttavia forzare un ordine di esecuzione differente nel seguente modo:

- Selezionare il blocco interessato
- Aprire il pannello dettagli
- Scegliere “MANUALE” come ordinamento
- Inserire un numero d'ordine, avendo cura di inserire un numero non ancora utilizzato

I blocchi con ordinamento manuale sono evidenziati nel seguente modo:



La figura seguente mostra un esempio di rete logica con un blocco in ordinamento manuale, ed evidenzia come modificare l'ordine di esecuzione dei blocchi:



Gli oggetti di IKON SERVER non hanno un ordine di esecuzione, perché essi non rappresentano una elaborazione da parte dell'unità logica, ma solo punti di lettura e scrittura dal campo; come già evidenziato in precedenza, gli stati dei nodi di uscita di tutti i blocchi (di tutti i programmi attivi) sono letti all'inizio di ogni ciclo di esecuzione, e i comandi ai nodi di ingresso di tutti i blocchi (di tutti i programmi attivi) vengono inviati su bus al termine del ciclo di esecuzione, a prescindere quindi dalla posizione dei blocchi nei programmi e dall'ordine dei programmi stessi.

4.11 PASSAGGIO DI VALORI TRA PROGRAMMI

Nonostante ogni programma sia una rete logica a sé stante, è possibile passare valori tra programmi differenti usando particolari blocchi logici detti “variabili”. Per creare una nuova variabile è necessario procedere nel seguente modo:

- Aprire la sezione “LOGICHE” del menu principale
- Identificare la sezione “VARIABILI BINARIE” (se si desidera creare una variabile di tipo ON/OFF) oppure “VARIABILI NUMERICHE”
- Premere il pulsante “+” corrispondente ed attendere che la nuova variabile sia inserita nella lista
- Selezionare la nuova variabile e trascinarla nel primo programma

E' possibile assegnare un nome alla variabile attraverso il pannello dettagli, in modo da identificarla più facilmente all'interno dei programmi in cui sarà usata.

Se si desidera assegnare alla variabile il valore di un nodo di uscita di un blocco logico, è sufficiente collegare quest'ultimo al nodo di ingresso (lato sinistro) della variabile; viceversa, per usare questo valore in altri programmi, collegare il nodo di uscita (lato destro) al nodo di ingresso di un altro blocco.

4.12 TIPI DI DATO

I nodi di ingresso e uscita dei blocchi possono prevedere due tipi di dato:

- BINARIO: sono ammessi solo i valori 1 (ON) e 0 (OFF)

- NUMERICO: è ammesso qualunque valore numerico, con eventuali restrizioni specifiche in base al blocco

Questi due tipi di dato sono incompatibili, pertanto l'editor impedisce la connessione di nodi binari con nodi numerici e viceversa: non appena si inizia il *drag&drop* per la creazione di un collegamento, i nodi incompatibili vengono resi semi-trasparenti, e non accettano il rilascio per la creazione della connessione.

5 LOGICHE

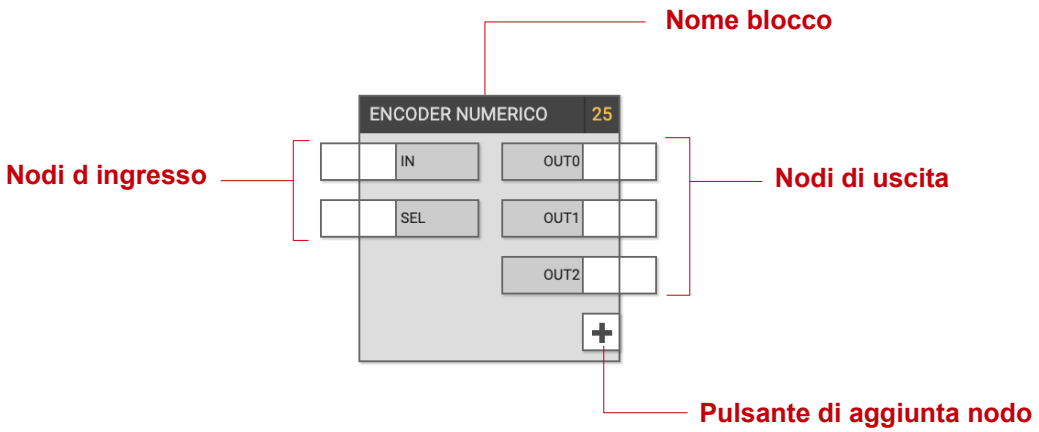
5.1 INTRODUZIONE

I blocchi logici permettono di effettuare operazioni su uno o più valori di ingresso, e restituiscono uno o più valori di uscita, che possono essere collegati con altri blocchi logici.

5.2 BLOCCHI LOGICI

5.2.1 LAYOUT

Come già anticipato, i blocchi logici si presentano graficamente come nell'esempio seguente:



5.2.2 NODI DI INGRESSO

I nodi di ingresso permettono di passare valori alle funzioni logiche. Selezionando un nodo di ingresso e aprendo il pannello dettagli, è possibile impostare le seguenti opzioni:

VALORE DI DEFAULT	Permette di impostare il valore del nodo da utilizzare all'inizio dell'esecuzione, finché non viene ricevuto un valore diverso, oppure se il nodo non è collegato ad un altro blocco

Il pannello dettagli, oltre alle opzioni suddette, mostra anche i valori possibili che il nodo può assumere; nel caso di nodi binari i valori possibili sono solo 0 (OFF) o 1 (ON), nel caso viceversa di nodi numerici i valori possibili dipendono dal tipo di nodo, e possono avere specifiche restrizioni.

5.2.3 NODI DI USCITA

I nodi di uscita restituiscono i risultati della funzione logica associata al blocco, e permettono di passarli ad altri blocchi.

Anche per i nodi di uscita è prevista la possibilità di impostare il **valore di default**.

5.2.4 AGGIUNTA E RIMOZIONE NODI

Alcuni blocchi prevedono un numero variabile di nodi; in questi casi, tipicamente il blocco, una volta trascinato dal menu laterale, contiene un set minimo di nodi, che può essere aumentato fino ad un numero massimo di nodi, premendo il pulsante “+”.

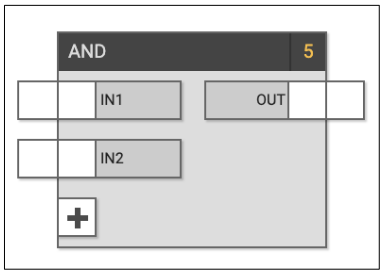
Per rimuovere un nodo precedentemente aggiunto, procedere come segue:

- Selezionare il nodo
- Aprire il pannello dettagli
- Premere il pulsante “ELIMINA”

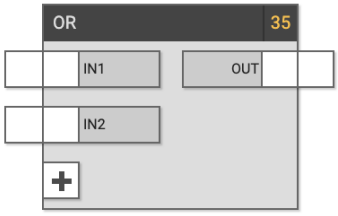
Le eventuali connessioni associate al nodo saranno cancellate.

5.3 LOGICHE COMBINATORIE

5.3.1 AND

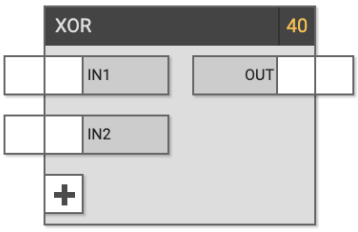
DESCRIZIONE:	Effettua la funzione logica AND tra due o più ingressi binari (fino ad un massimo di 10)			
ANTEPRIMA:				
NODI:	**TAG**	**DESCRIZIONE**	**IN**	**OUT**
	IN1 … IN10	Ingresso 1 … 10 Valori possibili: 0 → OFF 1 → ON	X	
	OUT	Uscita Valori possibili: 0 → OFF 1 → ON		X
	+	Aggiungi nodo	X	
OPZIONI:				

5.3.2 OR

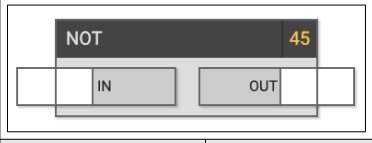
DESCRIZIONE:	Effettua la funzione logica OR tra due o più ingressi binari (fino ad un massimo di 10)			
ANTEPRIMA:				
NODI:	**TAG**	**DESCRIZIONE**	**IN**	**OUT**

	IN1 … IN10	Ingresso 1 … 10 Valori possibili: 0 → OFF 1 → ON	X	
	OUT	Uscita Valori possibili: 0 → OFF 1 → ON		X
	+	Aggiungi nodo	X	
OPZIONI:				

5.3.3 XOR

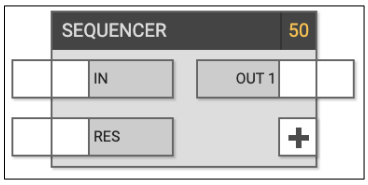
DESCRIZIONE:	Effettua la funzione logica XOR tra due o più ingressi binari (fino ad un massimo di 10)			
ANTEPRIMA:				
NODI:	**TAG**	**DESCRIZIONE**	**IN**	**OUT**
	IN1 … IN10	Ingresso 1 … 10 Valori possibili: 0 → OFF 1 → ON	X	
	OUT	Uscita Valori possibili: 0 → OFF 1 → ON		X
	+	Aggiungi nodo	X	
OPZIONI:				

5.3.4 NOT

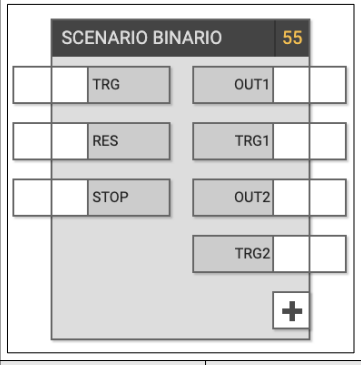
DESCRIZIONE:	Effettua la funzione logica NOT dell'ingresso			
ANTEPRIMA:				
NODI:	**TAG**	**DESCRIZIONE**	**IN**	**OUT**
	IN	Ingresso Valori possibili: 0 → OFF 1 → ON	X	
	OUT	Uscita Valori possibili: 0 → OFF 1 → ON		X
OPZIONI:				

5.4 SCENARI E SEQUENZE

5.4.1 SEQUENCER

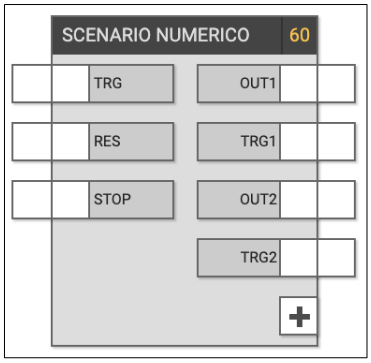
DESCRIZIONE:	Alla ricezione di un impulso nell'ingresso IN, attiva e disattiva in sequenza fino a 10 uscite di tipo booleano, tenendo ognuna di esse attiva per un tempo impostabile			
ANTEPRIMA:				
NODI:	**TAG**	**DESCRIZIONE**	**IN**	**OUT**
	IN	Inizio sequenza Valori possibili: 0 → OFF 1 → ON	X	
	RES	Reset sequenza Valori possibili: 0 → OFF 1 → ON	X	
	OUT1 … OUT10	Uscita 1 … 10 Valori possibili: 0 → OFF 1 → ON		X
	+	Aggiungi nodo		X
OPZIONI:	Sequenza ciclica	Stabilisce se la sequenza debba essere ripetuta una volta terminata Valori possibili: VERO/FALSO		
	Durata passo 1 … 10	Tempo di attesa tra il passo X ed il successivo Valori possibili: da 1 secondo a 12 ore		

5.4.2 SCENARIO BINARIO

DESCRIZIONE:	Alla ricezione di un impulso nell'ingresso TRG, esegue una sequenza di comandi di tipo booleano, ognuno impostabile, eventualmente intervallando ogni comando con un tempo prefissato, comune a tutte le uscite			
ANTEPRIMA:				
NODI:	**TAG**	**DESCRIZIONE**	**IN**	**OUT**

	TRG	Trigger ingresso Valori possibili: 0 → OFF 1 → ON	X	
	RES	Reset scenario Valori possibili: 0 → OFF 1 → ON	X	
	STOP	Stop scenario Valori possibili: 0 → OFF 1 → ON	X	
	OUT1 … OUT10	Uscita 1 … 10 Valori possibili: 0 → OFF 1 → ON		X
	TRG1 … TRG10	Trigger 1 … 10 Valori possibili: 0 → OFF 1 → ON		X
	+	Aggiungi nodo (e relativo trigger)		X
OPZIONI:	Intervallo uscite	Tempo di attesa tra il comando delle uscite Valori possibili: 1 … 60 (secondi)		
	Set uscita 1 … 10	Valore da impostare all'uscita 1 … 10 Valori possibili: 0 → Falso (OFF) 1 → Vero (ON)		

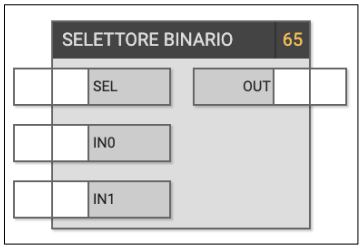
5.4.3 SCENARIO NUMERICO

DESCRIZIONE:	Alla ricezione di un impulso nell'ingresso TRG, esegue una sequenza di comandi di tipo numerico, ognuno impostabile, eventualmente intervallando ogni comando con un tempo prefissato, comune a tutte le uscite			
ANTEPRIMA:				
NODI:	**TAG**	**DESCRIZIONE**	**IN**	**OUT**
	TRG	Trigger ingresso Valori possibili: 0 → OFF 1 → ON	X	
	RES	Reset scenario Valori possibili: 0 → OFF	X	

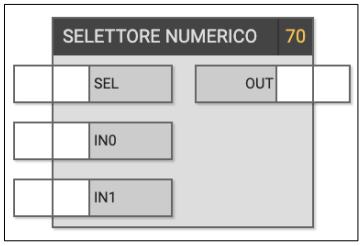
		1 → ON		
	STOP	Stop scenario Valori possibili: 0 → OFF 1 → ON	X	
	OUT1 … OUT10	Uscita 1 … 10 Valori possibili: qualunque valore numerico		X
	TRG1 … TRG10	Trigger 1 … 10 Valori possibili: 0 → OFF 1 → ON		X
	+	Aggiungi nodo + trigger		X
OPZIONI:	Intervallo uscite	Tempo di attesa tra il comando delle uscite Valori possibili: 1 … 60 (secondi)		
	Set uscita 1 … 10	Valore da impostare all'uscita 1 … 10 Valori possibili: qualunque valore numerico		

5.5 GATES

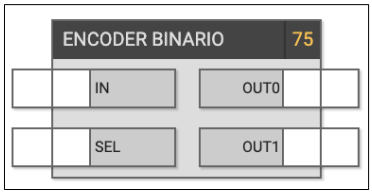
5.5.1 SELETTORE BINARIO

DESCRIZIONE:	Restituisce il valore di uno degli ingressi in base al valore dell'ingresso SEL che funge da selettore Se SEL=Falso → OUT=IN0 Se SEL=Vero → OUT=IN1			
ANTEPRIMA:				
NODI:	**TAG**	**DESCRIZIONE**	**IN**	**OUT**
	SEL	Selettore ingresso Valori possibili: 0 → OFF (in questo caso OUT = IN0) 1 → ON (in questo caso OUT = IN1)	X	
	IN0 IN1	Ingresso 0, ingresso 1 Valori possibili: 0 → OFF 1 → ON	X	
	OUT	Uscita Valori possibili: 0 → OFF 1 → ON		X
OPZIONI:				

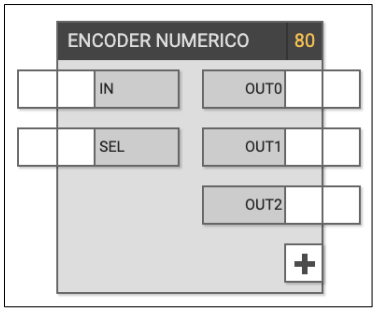
5.5.2 SELETTORE NUMERICO

DESCRIZIONE:	Restituisce il valore di uno degli ingressi in base al valore dell'ingresso SEL che funge da selettore Se SEL=Falso → OUT=IN0 Se SEL=Vero → OUT=IN1			
ANTEPRIMA:				
NODI:	**TAG**	**DESCRIZIONE**	**IN**	**OUT**
	SEL	Selettore ingresso Valori possibili: 0 → OFF (in questo caso OUT = IN0) 1 → ON (in questo caso OUT = IN1)	X	
	IN0 IN1	Ingresso 0, ingresso 1 Valori possibili: qualunque valore numerico	X	
	OUT	Uscita Valori possibili: qualunque valore numerico		X
OPZIONI:				

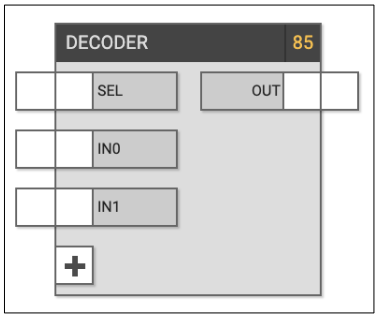
5.5.3 ENCODER BINARIO

DESCRIZIONE:	Pone il valore di IN (boolean) in una delle sue due uscite in base al valore di ingresso SEL (boolean) che funge da selettore Numero di uscite: 2			
ANTEPRIMA:				
NODI:	**TAG**	**DESCRIZIONE**	**IN**	**OUT**
	IN	Ingresso Valori possibili: 0 → OFF 1 → ON	X	
	SEL	Selettore uscita Valori possibili: 0 → IN viene impostato su OUT0 1 → IN viene impostato su OUT1	X	
	OUT0 OUT1	Uscita Valori possibili: 0 → OFF 1 → ON		X
OPZIONI:				

5.5.4 ENCODER NUMERICO

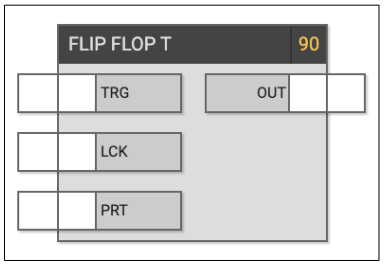
DESCRIZIONE:	Pone il valore di IN (boolean) in una delle sue uscite in base al valore di ingresso SEL (numerico) che funge da selettore. Numero di uscite: da 2 a 10			
ANTEPRIMA:				
NODI:	**TAG**	**DESCRIZIONE**	**IN**	**OUT**
	IN	Ingresso Valori possibili: 0 → OFF 1 → ON	X	
	SEL	Selettore uscita Valori possibili: 1 → IN viene posto su OUT0 2 → IN viene posto su OUT1 … 10 → IN viene posto su OUT9	X	
	OUT0 … OUT9	Uscita 0 … 9 Valori possibili: 0 → OFF 1 → ON		X
	+	Aggiungi uscita		X
OPZIONI:				

5.5.5 DECODER (O SELETTORE NUMERICO)

DESCRIZIONE:	Restituisce in uscita il valore di uno degli ingressi (boolean) in base al valore in ingresso SEL (numerico) che funge da selettore			
ANTEPRIMA:				
NODI:	**TAG**	**DESCRIZIONE**	**IN**	**OUT**
	SEL	Selettore ingresso Valori possibili: 0 → OUT viene impostato con IN0 1 → OUT viene impostato con IN1	X	

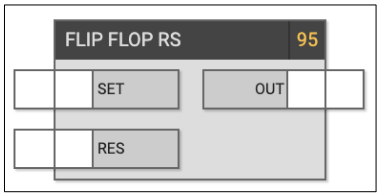
		… 9 → OUT viene impostato con IN9		
	IN0 … IN9	Ingresso 0 … 9 Valori possibili: 0 → OFF 1 → ON	X	
	OUT	Uscita Valori possibili: 0 → OFF 1 → ON		X
	+	Aggiungi ingresso	X	
OPZIONI:				

5.5.6 FLIP-FLOP T

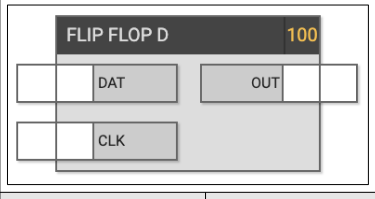
DESCRIZIONE:	Flip-flop di tipo T Funziona come un relè passo-passo. Ogni volta che viene proposto un fronte di salita al suo ingresso (TRG) l'uscita (OUT) cambia di stato. Se l'ingresso LCK (Blocco) è a 1 (Vero) viene inibito l'effetto del TRG per cui l'uscita non cambia mai. Se l'ingresso PRT (Priorità) è a 1 l'uscita assume il valore impostato nel parametro VAL (Valore prioritario). Può essere usato ad esempio per pilotare la luce di un corridoio. Si può fare in modo che la luce venga normalmente comandata solo se una soglia di luminosità è soddisfatta (tale condizione andrebbe a finire dentro il LCK) e sia comunque sempre accesa in orario notturno (Flag che andrebbe collegato all'ingresso PRT)			
ANTEPRIMA:				
NODI:	**TAG**	**DESCRIZIONE**	**IN**	**OUT**
	TRG	Trigger Valori possibili: 0 → OFF 1 → ON	X	
	LCK	Blocca lo stato attuale Valori possibili: 0 → OFF 1 → ON	X	
	PRT	Flag priorità Valori possibili: 0 → OFF 1 → ON	X	
	OUT	Segnale in uscita Valori possibili: 0 → OFF 1 → ON		X

OPZIONI:	Valore priorità	Valore da assegnare all'uscita in caso di flag prioritario Valori possibili: VERO / FALSO

5.5.7 FLIP-FLOP RS

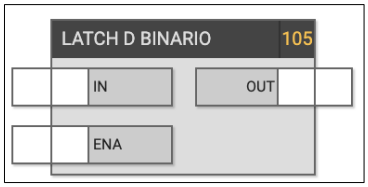
DESCRIZIONE:	Flip-flop di tipo RS E' un blocco di memoria elementare che viene "Caricata" con l'ingresso SET e resettata con l'ingresso RES (reset). Se entrambi gli ingressi sono a 1 prevale quello specificato dal parametro “priorità”. Può essere ad esempio usato per gestire una segnalazione di allarme. Un contatto di allarme va collegato al SET. Una volta che è andato ad 1 il Flip-Flop mantiene l'uscita a 1 fino a quando non viene resettata dal RES. In questo modo anche se l'allarme si "normalizza" (passa a 0) viene mantenuta l'informazione.			
ANTEPRIMA:				
NODI:	**TAG**	**DESCRIZIONE**	**IN**	**OUT**
	SET	Set Valori possibili: 0 → OFF 1 → ON	X	
	RES	Reset Valori possibili: 0 → OFF 1 → ON	X	
	OUT	Segnale in uscita Valori possibili: 0 → OFF 1 → ON		X
OPZIONI:	Priorità selezione	Valori possibili: Set / Reset		

5.5.8 FLIP-FLOP D

DESCRIZIONE:	Flip-flop di tipo D Il funzionamento è simile a quello del Latch D con la differenza che il Filp-flop D opera sulla variazione del fronte di CLK. Il dato in DAT viene riportato in OUT solo in corrispondenza del fronte di salita del segnale CLK e lo mantiene fino al successivo fronte di salita di CLK (di fatto, questo blocco costituisce una cella di memoria)			
ANTEPRIMA:				
NODI:	**TAG**	**DESCRIZIONE**	**IN**	**OUT**
	DAT	Dato	X	

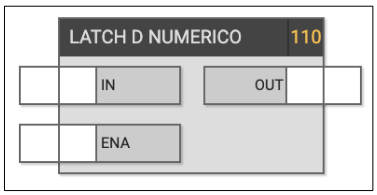
		Valori possibili: 0 → OFF 1 → ON		
	CLK	Clock Valori possibili: 0 → OFF 1 → ON	X	
	OUT	Segnale in uscita Valori possibili: 0 → OFF 1 → ON		X
OPZIONI:	Priorità selezione	Valori possibili: DAT / CLK		

5.5.9 LATCH D BINARIO

DESCRIZIONE:	Latch D Binario In questo blocco il segnale in ingresso IN viene propagato in uscita OUT se il segnale di enable ENA è abilitato (1). Se il segnale di enable ENA è disabilitato, in uscita OUT permane l'ultimo stato. Quando enable ENA ritorna attivo quindi (transizione 0 --> 1), in uscita OUT viene inviato l'ultimo valore letto nel nodo di ingresso IN. In sostanza, con ENA=0, il blocco Latch memorizza l'ultimo valore letto per inviarlo in uscita nel momento il cui ENA ritorna ad essere abilitato. Il formato dati di IN e OUT è binario			
ANTEPRIMA:				
NODI:	**TAG**	**DESCRIZIONE**	**IN**	**OUT**
	IN	Ingresso Valori possibili: 0 → OFF 1 → ON	X	
	ENA	Enable Valori possibili: 0 → OFF 1 → ON	X	
	OUT	Segnale in uscita Valori possibili: 0 → OFF 1 → ON		X
OPZIONI:	Priorità selezione	Valori possibili: In / Ena		

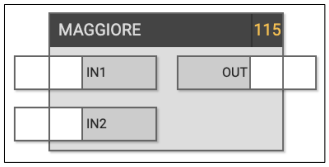
5.5.10 LATCH D NUMERICO

DESCRIZIONE:	Latch D NUMERICO In questo blocco il segnale in ingresso IN viene propagato in uscita OUT se

	il segnale di enable ENA è abilitato (1). Se il segnale di enable ENA è disabilitato, in uscita OUT permane l'ultimo stato. Quando enable ENA ritorna attivo quindi (transizione 0 --> 1), in uscita OUT viene inviato l'ultimo valore letto nel nodo di ingresso IN. In sostanza, con ENA=0, il blocco Latch memorizza l'ultimo valore letto per inviarlo in uscita nel momento il cui ENA ritorna ad essere abilitato. Il formato dati di IN e OUT è numerico			
ANTEPRIMA:				
NODI:	**TAG**	**DESCRIZIONE**	**IN**	**OUT**
	IN	Ingresso Valori possibili:qualunque valore numerico	X	
	ENA	Enable Valori possibili: 0 → OFF 1 → ON	X	
	OUT	Segnale in uscita Valori possibili: 0 → OFF 1 → ON		X
OPZIONI:	Priorità selezione	Valori possibili: In / Ena		

5.6 CONFRONTI

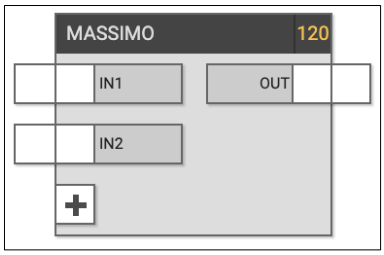
5.6.1 OPERATORI DI CONFRONTO

DESCRIZIONE:	Confronta il valore dei due ingressi, e restituisce in uscita un valore VERO/FALSO in base all'operatore specifico Operatori disponibili: • Maggiore • Maggiore uguale • Minore • Minore uguale • Uguale • Diverso			
ANTEPRIMA:				
NODI:	**TAG**	**DESCRIZIONE**	**IN**	**OUT**
	IN1 IN2	Ingresso 1, ingresso 2 Valori possibili: qualunque valore numerico	X	

	OUT	Risultato confronto Valori possibili: 0 → OFF 1 → ON		X
OPZIONI:				

5.7 OPERAZIONI

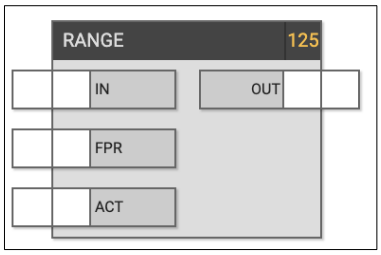
5.7.1 OPERATORI MATEMATICI

DESCRIZIONE:	Effettua una operazione matematica sugli ingressi, in base alla tipologia di operatore Operatori disponibili: • Massimo • Minimo • Media • Somma • Sottrazione • Moltiplicazione • Divisione • Valore assoluto • Log10 • Integratore			
ANTEPRIMA:				
NODI:	**TAG**	**DESCRIZIONE**	**IN**	**OUT**
	IN1 IN2 (*) ...	Ingresso 1, ingresso 2 ... Valori possibili: qualunque valore numerico	X	
	OUT	Valore (risultato dell'operazione) Valori possibili: qualunque valore numerico		X
OPZIONI:				

(*) Il numero delle uscite può essere limitato in base all'operazione (es: divisione max 2, valore assoluto max 1)

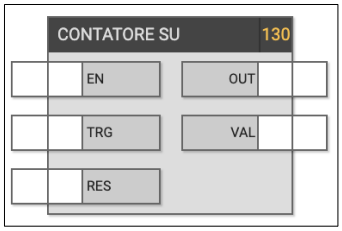
5.7.2 RANGE

DESCRIZIONE:	Effettua una interpolazione lineare del valore di ingresso IN in base ad una mappatura assegnata, detta anche “caratteristica”, definita da due coppie di valori (X,Y). Il valore IN viene rapportato tra X0 e X1, e questo rapporto viene a sua volta calcolato tra i valori Y0 e Y1 per determinare il valore in uscita. Se viene impostata la modalità prioritaria, viene restituito un valore

	prefissato. Il tipico campo di applicazione di questo blocco è la conversione di valori tra grandezze diverse.			
ANTEPRIMA:				
NODI:	**TAG**	**DESCRIZIONE**	**IN**	**OUT**
	IN	Ingresso Valori possibili: qualunque valore numerico	X	
	FPR	Abilitazione priorità Valori possibili: 0 → No priorità 1 → Priorità (viene restituito il valore prioritario)	X	
	ACT	Funzionamento diretto/inverso Valori possibili: 0 → Funzionamento diretto 1 → Funzionamento inverso	X	
	OUT	Uscita Valori possibili: qualunque valore numerico		X
OPZIONI:	X0 Y0 X1 Y1	Caratteristiche dell'interpolazione lineare Valori possibili: qualunque valore numerico		
	Valore prioritario	Valore da restituire in caso di abilitazione priorità		

5.8 CONTATORI

5.8.1 CONTATORE SU, CONTATORE GIÙ, CONTATORE SU/GIÙ

DESCRIZIONE:	Conta il numero di impulsi ricevuti in ingresso (trigger), incrementando o decrementando ogni volta il proprio valore (in base al tipo di contatore). Tipi di contatore: contatore su, contatore giù, contatore su/giù			
ANTEPRIMA:				
NODI:	**TAG**	**DESCRIZIONE**	**IN**	**OUT**
	EN	Abilitazione Valori possibili: 0 → Non abilitato 1 → Abilitato	X	

	TRG	Trigger Valori possibili: 0 → OFF 1 → ON (viene incrementato il contatore)	X	
	RES	Reset Valori possibili: 0 → OFF 1 → ON (resetta il contatore)	X	
	OUT	Uscita Valori possibili: 0 → OFF 1 → ON		X
	VAL	Valore attuale Valori possibili: qualunque valore numerico		X
OPZIONI:	Preset	Valore predefinito, impostato quando viene effettuato il reset oppure all'avvio della logica Valori possibili: qualunque valore numerico		

5.9 TIMERS

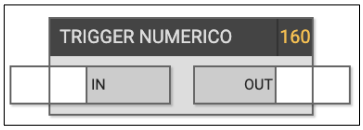
5.9.1 TIMER

DESCRIZIONE:	Ritarda di un tempo preimpostato il valore ricevuto in ingresso Quando viene ricevuto un 1 nell'ingresso IN (fronte di salita), parte un contatore interno fino al tempo specificato come “ritardo in salita”, dopo di che l'uscita viene portata ad 1; viceversa, alla ricezione di uno 0 in ingresso (fronte di discesa), il blocco aspetta il tempo specificato come “ritardo in discesa” prima di porre l'uscita a 0.			
ANTEPRIMA:				
NODI:	**TAG**	**DESCRIZIONE**	**IN**	**OUT**
	IN	Segnale ON/OFF in ingresso Valori possibili: 0 → OFF 1 → ON	X	
	OUT	Segnale ON/OFF in uscita, ritardato dal timer Valori possibili: 0 → OFF 1 → ON		X
OPZIONI:	Ritardo in salita	Ritardo nella propagazione del fronte di salita ricevuto in ingresso Valori possibili: da 1 secondo a 24 ore		
	Ritardo in discesa	Ritardo nella propagazione del fronte di discesa ricevuto in ingresso Valori possibili: da 1 secondo a 24 ore		

5.9.2 TRIGGER

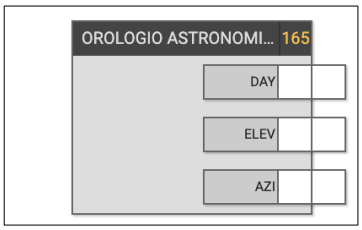
DESCRIZIONE:	Genera un trigger (impulso della durata di un ciclo) su un fronte rilevato in ingresso Quando riceve un 1 in ingresso, imposta ad 1 l'uscita per la durata di un singolo ciclo di elaborazione, poi l'uscita viene nuovamente posta a 0. in questo modo è possibile generare un “impulso” per blocchi logici che lo richiedono (es: scenari, sequencer etc...) sul fronte di salita dell'ingresso.			
ANTEPRIMA:				
NODI:	**TAG**	**DESCRIZIONE**	**IN**	**OUT**
	IN	Fronte in ingresso	X	
	OUT	Impulso della durata di un ciclo		X
OPZIONI:	Fronte	Fronte di salita o discesa da rilevare in ingresso		

5.9.3 TRIGGER NUMERICO

DESCRIZIONE:	Genera un trigger (impulso della durata di un ciclo) su un fronte rilevato in ingresso Quando riceve un determinato valore (numerico) in ingresso, imposta ad 1 l'uscita per la durata di un singolo ciclo di elaborazione, poi l'uscita viene nuovamente posta a 0. in questo modo è possibile generare un “impulso” per blocchi logici che lo richiedono (es: scenari, sequencer etc...) alla ricezione di un preciso valore in ingresso.			
ANTEPRIMA:				
NODI:	**TAG**	**DESCRIZIONE**	**IN**	**OUT**
	IN	Fronte in ingresso	X	
	OUT	Impulso della durata di un ciclo		X
OPZIONI:	Condizione	Valore ricevuto in ingresso a fronte del quale viene generato il trigger in uscita		

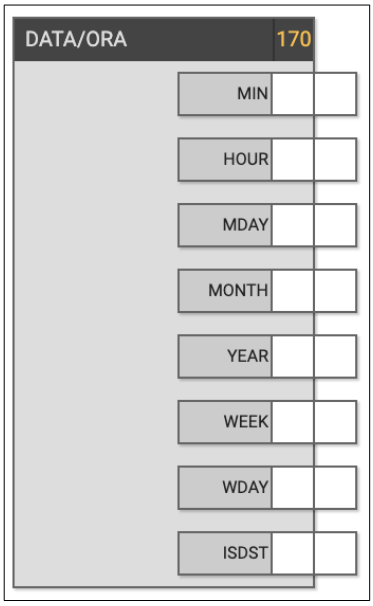
5.9.4 OROLOGIO ASTRONOMICO

DESCRIZIONE:	Restituisce in uscita crepuscolare, elevazione e azimut in ogni momento del giorno in base alle coordinate geografiche impostate. Permette di pilotare funzionalità domotiche (es: oscuranti) in base alla posizione del sole.

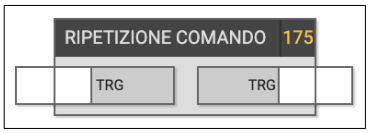
ANTEPRIMA:				
NODI:	**TAG**	**DESCRIZIONE**	**IN**	**OUT**
	DAY	Crepuscolare Valori possibili: 0 → Notte 1 → Giorno		
	ELEV	Elevazione del sole; restituisce l'altezza del sole rispetto all'orizzonte. Il valore 0 indica l'orizzonte; valori positivi indicano giorno, negativi notte. Valori possibili: -90 … +90		X
	AZI	Azimut Restituisce la distanza angolare del sole rispetto al NORD Il valore 0 indica NORD, 90 EST, 180 SUD e 270 OVEST Valori possibili: 0 … 359		
OPZIONI:	Latitudine Longitudine	Coordinate geografiche Valori possibili: -90 … +90 (latitudine), -180...+180 (longitudine) Valore numerico con punto come separatore decimale, massimo 7 cifre decimali di precisione		
	Soglia	Soglia relativa all'elevazione del sole per determinare l'uscita giorno/notte (valore di default 0°, in questo caso quando il sole supererà l'orizzonte l'uscita DAY sarà giorno)		
	Fuso Orario	Selezionare il fuso appropriato dal menu a tendina		
	Cambio ora	Criterio di cambio ora solare / legale Valori possibili: disabilitato, automatico e manuale In caso di manuale, viene chiesto di specificarlo nella apposita opzione		

5.9.5 DATA/ORA

DESCRIZIONE:	Restituisce l'ora corrente del sistema

ANTEPRIMA:				
NODI:	**TAG**	**DESCRIZIONE**	**IN**	**OUT**
	MIN HOUR	Ore e minuti che compongono ora corrente		X
	MDAY MONTH YEAR	Componenti della data (giorno del mese, mese e anno)		X
	WEEK	Numero della settimana		X
	WDAY	Giorno della settimana		X
	ISDST	Specifica se è attiva l'ora legale (1) o no (0)		X
OPZIONI:				

5.9.6 RIPETIZIONE COMANDO

DESCRIZIONE:	E' provvisto di due nodi trigger; all'arrivo di un trigger sull'ingresso TRG genera N+1 trigger sul nodo TRG di uscita intervallati uno dall'altro da un parametro configurabile.			
ANTEPRIMA:				
NODI:	**TAG**	**DESCRIZIONE**	**IN**	**OUT**
	TRG	Trigger in ingresso	X	
	TRG	Trigger in uscita, ripetuto per n volte con intervallo impostato nelle opzioni		X
OPZIONI:	Intervallo (s)	Tempo in secondi che intercorre tra la generazione di un trigger in uscita ed il successivo		
	Ripetizioni	Numero di trigger da generare oltre al primo		

5.9.7 OSCILLATORE

DESCRIZIONE:	Genera un trigger (impulso della durata di un ciclo) ciclicamente in base ad un tempo impostato nelle opzioni.

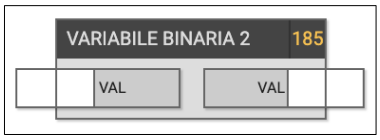
	Funge di fatto da *clock* per operazioni che devono essere effettuate ciclicamente sulla base di un intervallo di tempo prefissato; può essere ad esempio collegato al nodo TRG di uscita di uno o più oggetti, per forzare l'emissione periodica del valore sul campo (anche se non è cambiato)			
ANTEPRIMA:				
NODI:	**TAG**	**DESCRIZIONE**	**IN**	**OUT**
	TRG	Impulso della durata di un ciclo		X
OPZIONI:	Timer	Tempo tra l'emissione di un trigger ed il successivo		

5.10 VARIABILI

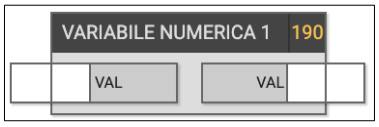
5.10.1 PREMESSA

Come illustrato nella sezione 4.11, le variabili permettono di passare valori tra programmi differenti. Le variabili devono essere precedentemente create attraverso il pulsante "+" nella apposita sezione del menu principale, quindi possono essere trascinate nei programmi che le devono utilizzare.

5.10.2 VARIABILI BINARIE

DESCRIZIONE:	Permettono di trasferire un valore booleano tra programmi differenti.			
CATEGORIA:	Variabili binarie			
ANTEPRIMA:				
NODI:	**TAG**	**DESCRIZIONE**	**IN**	**OUT**
	VAL	Valore da assegnare alla variabile Valori possibili: 0 → OFF 1 → ON	X	
	VAL	Valore corrente della variabile Valori possibili: 0 → OFF 1 → ON		X
OPZIONI:				

5.10.3 VARIABILI NUMERICHE

DESCRIZIONE:	Permettono di trasferire un valore booleano tra programmi differenti.			
CATEGORIA:	Variabili binarie			
ANTEPRIMA:				
NODI:	**TAG**	**DESCRIZIONE**	**IN**	**OUT**

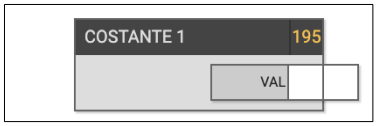
	VAL	Valore da assegnare alla variabile Valori possibili: qualunque valore numerico	X	
	VAL	Valore corrente della variabile Valori possibili: qualunque valore numerico		X
OPZIONI:				

5.11 COSTANTI

5.11.1 PREMESSA

Le costanti possono essere impiegate per impostare un valore in ingresso ad un blocco, con un valore che non può variare durante l'esecuzione. Analogamente a quanto visto per le variabili, le costanti devono essere create premendo il pulsante [+] nella toolbar, prima di poter essere trascinate nel task.

5.11.2 COSTANTI

DESCRIZIONE:	Permettono di impostare un valore numerico fisso collegandolo ad altri blocchi logici			
CATEGORIA:	Costanti			
ANTEPRIMA:				
NODI:	**TAG**	**DESCRIZIONE**	**IN**	**OUT**
	VAL	Valore della costante (numerico)		X
OPZIONI:				

6 TRIGGER

6.1 INTRODUZIONE

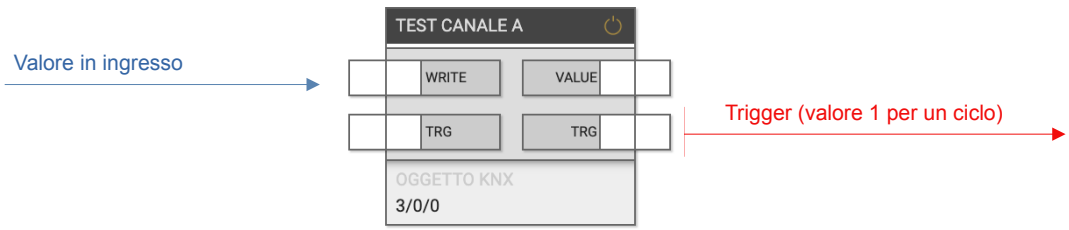
In condizioni normali, il modulo logico mantiene lo stato degli oggetti di uscita in base allo stato della logica; non appena questi oggetti cambiano di stato (ad esempio, perché ricevono un comando dal bus, esterno alla logica), il loro stato viene immediatamente ripristinato a quello voluto dalla logica.

Non sempre questo comportamento è desiderabile; in molte situazioni, infatti, si vuole comandare un oggetto in base al cambio di stato di altri oggetti, ma se poi questo oggetto viene comandato esternamente, lo si vuole mantenere in quest'ultimo stato. In altri termini, si vuole creare una logica che ragioni “ad eventi”.

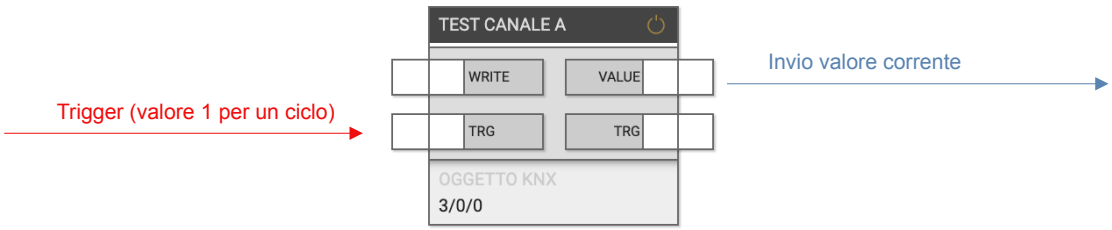
In questo caso, è necessario utilizzare i nodi TRIGGER di ingresso ed uscita.

6.2 TRIGGER DI COMANDO E DI STATO

I **trigger di stato (lato destro del blocco)** emettono un impulso (ovvero un valore 1 per un singolo ciclo di esecuzione dello script, poi tornano a 0) ogni volta che viene ricevuto un valore su un determinato oggetto.



I **trigger di comando (lato sinistro del blocco)**, viceversa, permettono di stabilire quando emettere un comando su un oggetto; ogni qualvolta viene mandato un impulso (valore 1) su questo trigger, il valore dell'oggetto viene mandato sul bus, anche se non è cambiato dal ciclo di esecuzione precedente.

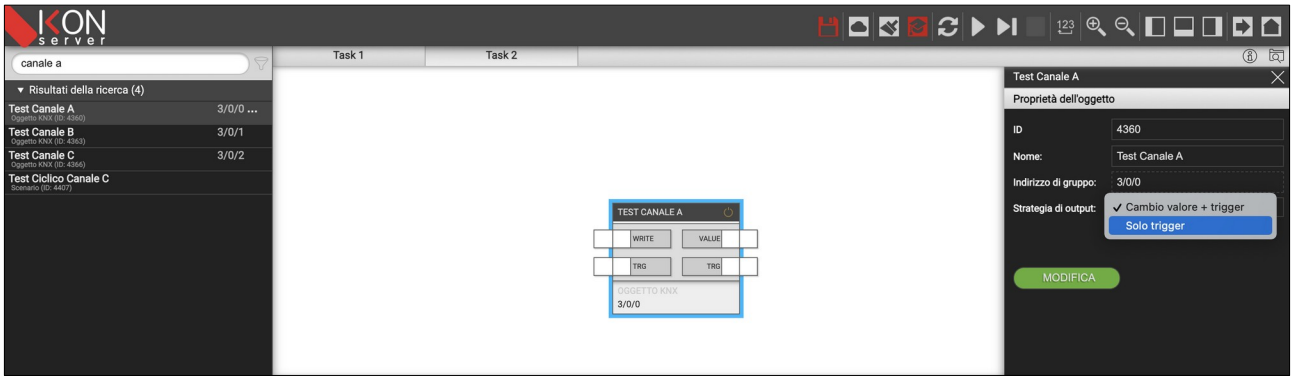


Nota: i trigger di stato trasmettono un impulso solo se il corrispondente nodo è connesso ad una logica.

6.3 STRATEGIA DI OUTPUT

Come detto in precedenza, in condizioni normali, i nodi di uscita di un oggetto all'interno di un task, vengono comandati in base al **cambio di valore** della logica a monte (o dell'oggetto rispetto al valore calcolato dalla logica a monte); inoltre, se viene ricevuto un impulso sul **trigger di comando** (nodo TRG sul lato sinistro del blocco) il valore viene nuovamente inviato in uscita, anche se non è cambiato.

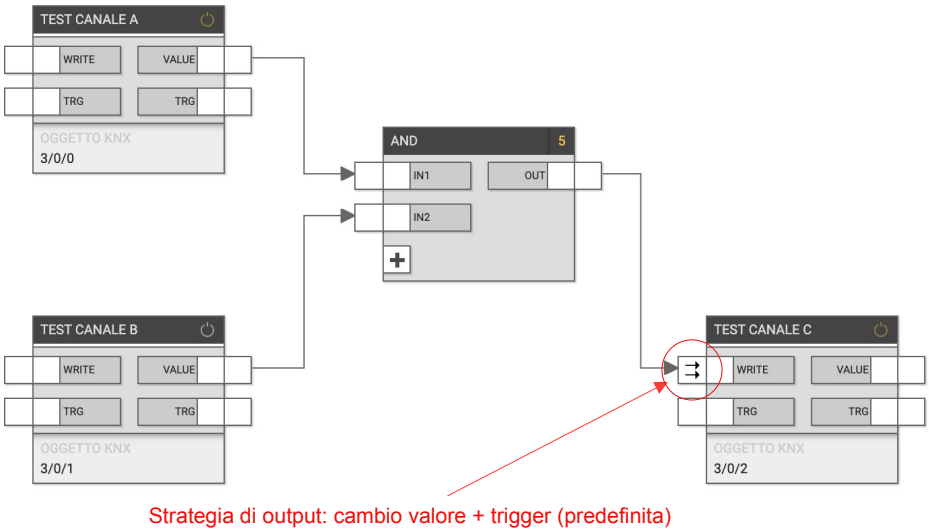
E' tuttavia possibile impostare i nodi di uscita in modo da inviare il valore **solo su trigger** modificando il selettore **strategia di output** nel pannello dei dettagli, dopo aver selezionato l'oggetto in questione:



In questo modo, solo se viene ricevuto un impulso viene comandato il corrispondente valore; in caso contrario, non viene mandato nessun comando, sia che la logica a monte cambi, sia che cambi lo stato dell'oggetto per motivi esterni alla logica (es: comando da parte dell'utente).

6.4 CONNESSIONI TRIGGER

Si prenda a questo punto ad esempio la seguente semplice logica:



Come evidenziato, l'oggetto di uscita ha come strategia di output "cambio valore + trigger" il che significa che:

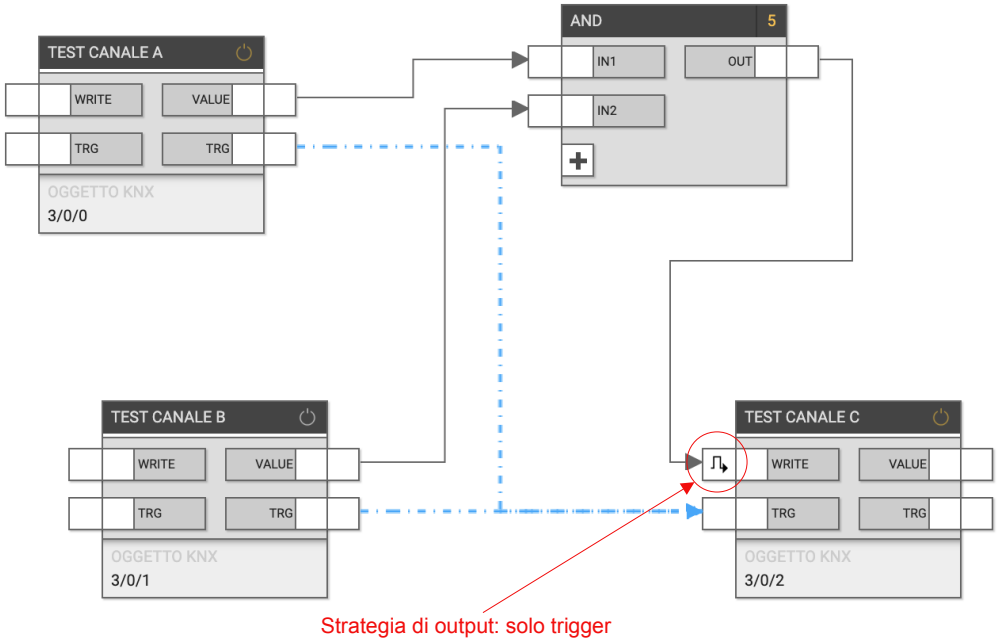
- Se cambia uno degli ingressi (CANALE A o B) l'oggetto CANALE C viene comandato con il valore della AND tra i due ingressi

- Se cambia il valore di CANALE C per motivi esterni alla logica (es: comando manuale da parte dell'utente, comando dal campo etc…) esso viene istantaneamente ri-allineato al valore della logica AND

Se si desidera viceversa che CANALE C venga comandato **solo quando cambia uno degli ingressi della logica** ma lasciare viceversa la libertà di poterlo comandare ad un valore differente a livello manuale, è necessario:

- Impostare la sua strategia di output a **solo trigger**
- Collegare il suo **trigger di comando** ai **trigger di stato** degli oggetti CANALE A e B

Per rendere fattibile questo tipo di collegamento con più trigger di ingresso, i nodi TRG di comando possono essere connessi a più ingressi, a differenza dei normali nodi di uscita (che viceversa accettano una sola connessione in ingresso all'interno dello stesso task).



Quando si collega il trigger di comando, la strategia di output dell'oggetto viene automaticamente impostata a "solo trigger" (ed il simbolo sui nodi di uscita cambia di conseguenza).

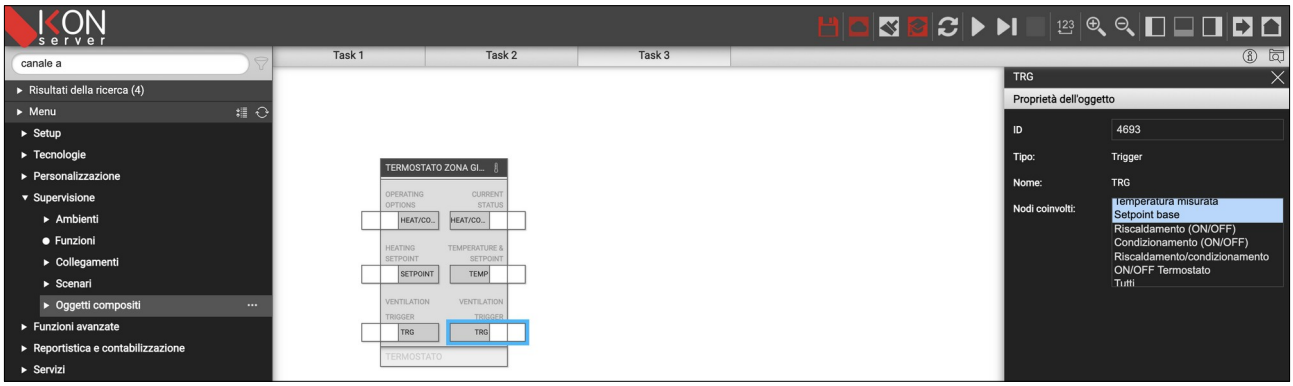
E' comunque possibile impostare la strategia manualmente, per regolamentare quando debba essere inviato il comando sul bus; il trigger infatti può essere usato anche per inviare periodicamente il valore di un'uscita, in aggiunta al suo cambio di stato, collegandovi a monte una logica periodica, un po' come avviene con molti dispositivi KNX.

6.5 TRIGGER E COMPOSITI

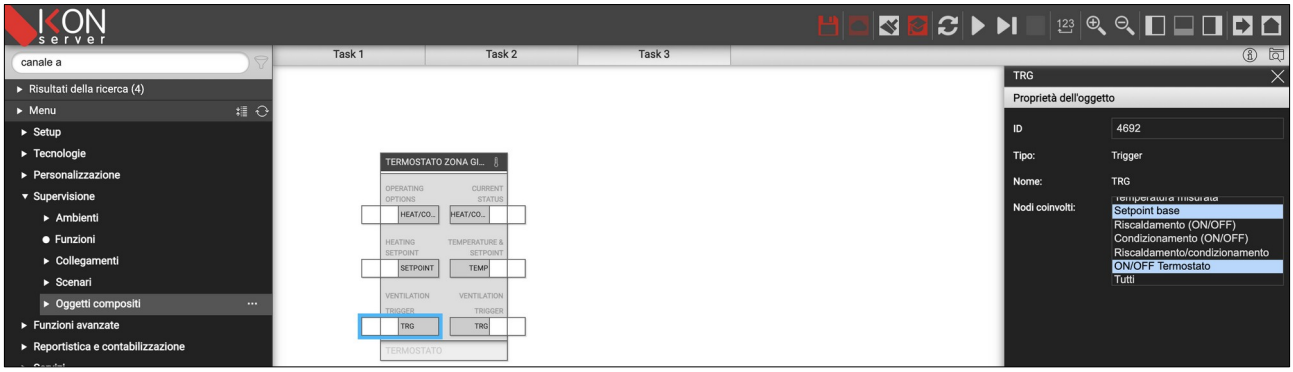
Nel caso di oggetti compositi, è disponibile un solo nodo TRG cumulativo, e non uno per ogni singolo sotto-nodo. In questo caso, è possibile stabilire a quali sotto-oggetti faccia riferimento il trigger, attraverso il pannello di dettaglio (dopo aver selezionato il trigger).

Nel caso di **trigger di stato**, è possibile selezionare quali sotto-oggetti determinano l'impulso al cambio di stato, attraverso il selettore **nodi interessati**; per impostazione predefinita, sono attivi

tutti i sotto-oggetti (quindi ogni qualvolta uno di essi cambia di stato, viene generato un impulso nella logica LUA), ma è possibile selezionare solo un sottoinsieme di essi:



In modo analogo, nel caso dei **trigger di comando**, è possibile stabilire quali sotto-oggetti debbano essere comandati quando si riceve un impulso:



Nota: come già evidenziato in precedenza, i trigger TRG trasmettono un impulso solo se i corrispondenti nodi sono connessi ad una logica.

7 CONNESSIONI

7.1 INTRODUZIONE

Come visto in precedenza, per realizzare un task logico, è necessario collegare i nodi di ingresso e di uscita dei diversi blocchi. Questo capitolo approfondisce le tipologie di connessione, le relative opzioni, e l'impatto che esse hanno sul codice LUA alla base di un programma logico.

7.2 TIPI DI CONNESSIONE

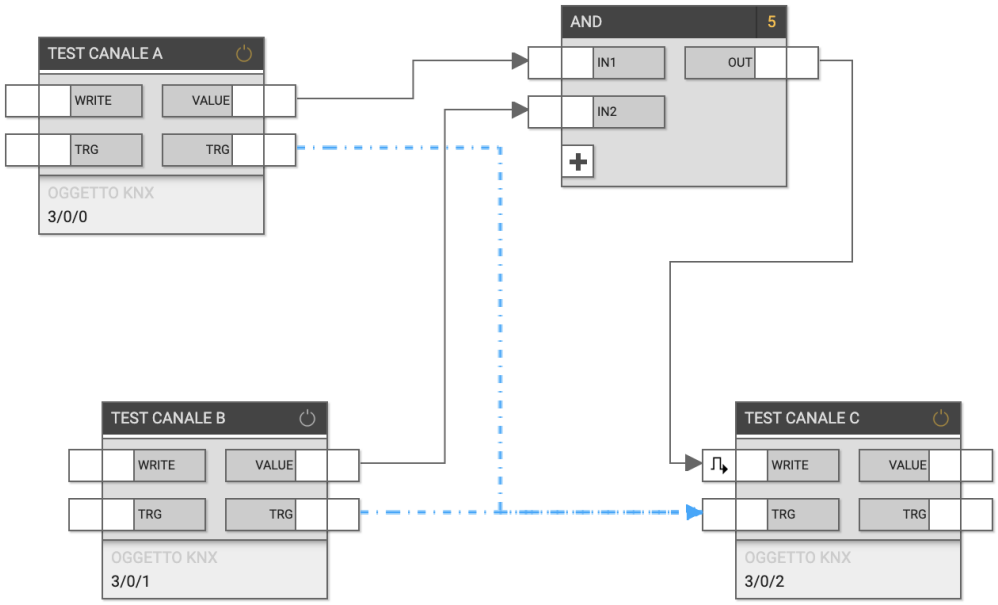
A seconda del tipo di nodo che connettono, le connessioni possono essere di due tipologie:

- Connessioni standard
- Connessioni trigger

Le prime collegano nodi di ingresso ed uscita dei blocchi, siano essi oggetti di IKON SERVER o blocchi logici di libreria, passando valori di tipo boolean o numerico (a seconda del tipo di blocco logico interessato). Questo tipo di connessione rappresenta il passaggio di un valore tra un nodo di ingresso di un blocco (lato destro del blocco nell'editor grafico) ed un nodo di uscita di un altro blocco (lato sinistro del blocco) durante l'esecuzione del programma logico.

Le connessioni di tipo trigger, viceversa, passano un impulso della durata di un ciclo di esecuzione, quando viene generato dal nodo TRG di ingresso di un oggetto di IKON SERVER; solitamente questo impulso viene collegato ad uno o più nodi TRG di altri oggetti, per comandarli solo su cambio di stato degli ingressi della logica (come meglio approfondito nel capitolo dedicato ai TRIGGER) ma può essere utilizzato come ingresso di una qualunque rete logica, assieme ad altri trigger oppure a valori boolean, interessando anche blocchi logici.

Le connessioni che interessano nodi trigger in ingresso, sono rappresentate nell'editor con un tratto linea-punto, come evidenziato nella figura seguente:

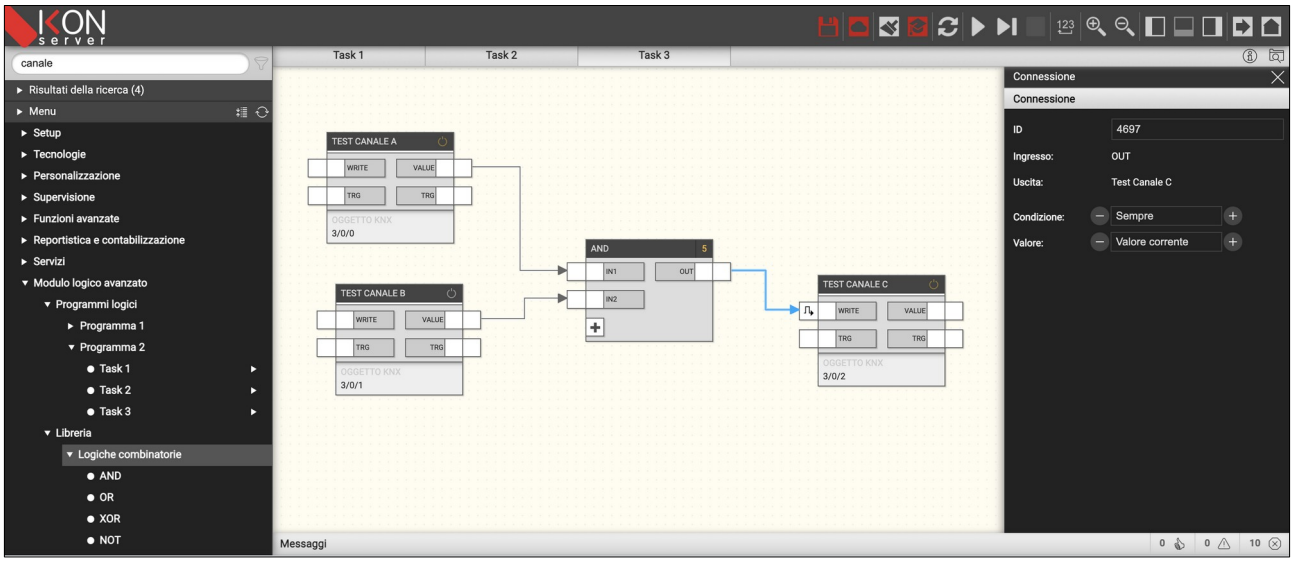


7.3 CONDIZIONI SULLE CONNESSIONI

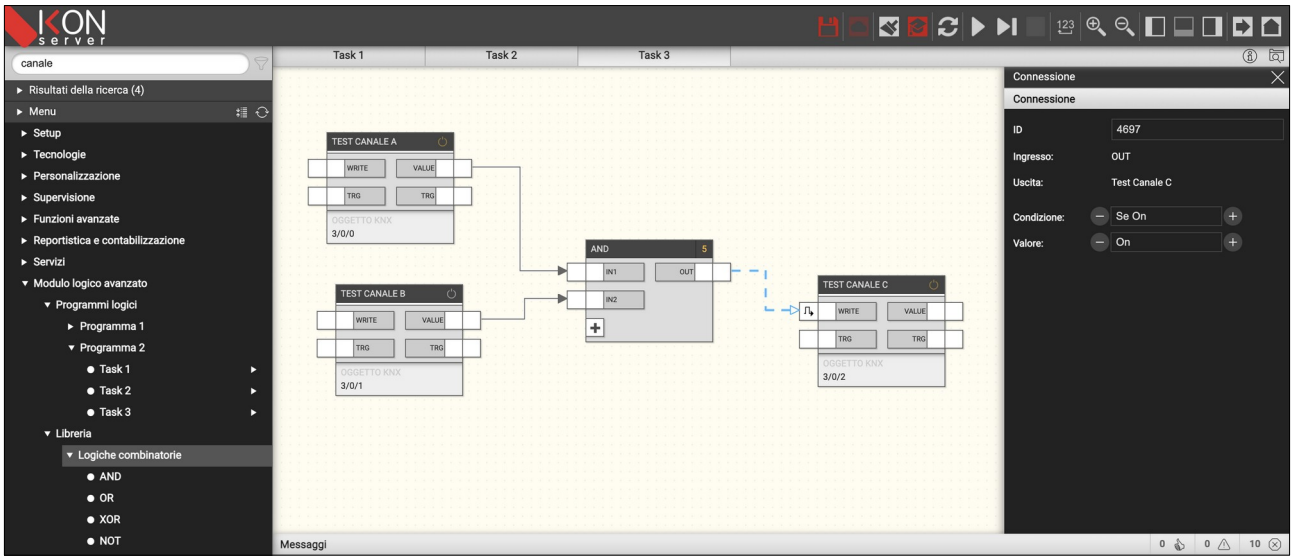
A partire dalla versione 2.9 di IKON SERVER, è possibile impostare una condizione su alcune tipologie di connessione, vincolando l'esecuzione del comando solo ad alcuni valori assunti dalla logica a monte della connessione. Questo tipo di filtro è disponibile solo sulle **connessioni di comando degli oggetti** di IKON SERVER e non le connessioni verso blocchi logici di libreria.

Selezionando una connessione che interessa, come nodo di destinazione, un oggetto di IKON SERVER, e aprendo il pannello laterale, è possibile modificare i seguenti parametri:

CONDIZIONE	Permette di stabilire se il comando del nodo di uscita debba essere eseguito **sempre** (qualunque sia il valore) oppure solo nel caso in cui la logica a monte assuma uno specifico valore, da selezionare nell'apposito menu a tendina
VALORE	Permette di stabilire quale valore debba essere inviato come comando all'oggetto di destinazione. Il valore predefinito è "valore corrente" (ovvero il valore della logica a monte) ma è possibile specificare un valore fisso tra quelli disponibili nell'apposito menu a tendina; in questo caso, viene passato sempre quel valore, a prescindere dal risultato della logica a monte



Impostando un filtro differente da quello predefinito, la connessione viene evidenziata con un tratteggio ed una freccia di destinazione vuota (invece che piena), come nella figura seguente:



In questo esempio, l'oggetto CANALE C viene comandato con un ON solo nel caso in cui la logica AND sia verificata. In tutti gli altri casi, non viene effettuato nessun comando sull'oggetto in questione, e quindi esso può assumere valori anche differenti da quello previsto dalla logica (nel caso specifico, può andare ad ON), in quanto vengono ignorati dal programma logico.

7.4 CONNESSIONI E CODICE LUA

La funzione delle connessioni all'interno del codice LUA generato dall'editor, è di passare il valore di un nodo ad un altro nodo; ogni nodo è una entità nel database di IKON SERVER (identificata con un ID univoco) e ad esso corrisponde una variabile LUA nello script, pertanto una connessione si traduce in una assegnazione di valore da una variabile (che rappresenta un nodo) ad un'altra:

```
[…]
            -- START TASK "Task 2" [4685]

            -- luafunction "andn" [4686]
            v4687=v4360                         --- Relation 10896
            v4688=v4363                         --- Relation 10897
            v4689=dlab.andn(4686,2,v4687,v4688)
            t4366=t4360 or t4363
            v4366=v4689                         --- Relation 10898

            -- Swap input vars to output vars
            o4366=v4366
            -- END TASK
[...]
```

Nel codice LUA, un commento a fianco di queste assegnazione di variabile, riporta il riferimento alla relazione all'interno del database; questa informazione può essere utile in caso di diagnostica da parte dell'assistenza tecnica, qualora il programma logico non si comporti come ci si aspetta.

Per ulteriori dettagli, si rimanda al capitolo dedicato alla struttura del codice LUA generato dall'editor, più avanti in questo manuale.

8 SIMULAZIONE ED ESECUZIONE

8.1 INTRODUZIONE

Una volta realizzato un task logico, è possibile simularne il funzionamento all'interno dell'editor, inserendo manualmente lo stato degli ingressi e verificando in tempo reale l'elaborazione delle uscite, anche ad opera dei blocchi logici che comportano una variazione delle uscite nel tempo.

Una volta che si è sicuri della correttezza di un programma logico, lo si può mettere in esecuzione; da quel momento, anche a fronte di eventuale riavvio del sistema, il programma viene automaticamente avviato in background.

8.2 TIPI DI SIMULAZIONE

Sono disponibili due tipologie di simulazione:

- **Simulazione continua**: l'esecuzione dei task avviene in background e risente in *real-time* dei cambi di stato dei nodi
- **Simulazione passo-passo**: ogni ciclo di esecuzione dei task deve essere lanciato manualmente, tra uno e l'altro è possibile modificare lo stato dei nodi

La prima tipologia permette una valutazione più realistica delle reti logiche realizzate, la seconda consente una verifica approfondita e puntuale di ogni singolo passaggio di valori tra blocchi, ed offre un livello di diagnostica più elevato.

8.3 AMBIENTE GRAFICO DI SIMULAZIONE

Premendo uno dei pulsanti di simulazione (continua o passo-passo), la finestra dell'editor subisce le seguenti modifiche:

- Il menu principale viene limitato alla sola vista PROGETTO, permettendo solo il passaggio tra programmi logici. Non è possibile creare o cancellare programmi.
- Il pannello dettagli viene chiuso per offrire il massimo spazio di lavoro per la simulazione
- Ogni operazione di drag&drop, collegamento, modifica o cancellazione del contenuto dei programmi logici è bloccato
- I nodi assumono una colorazione in base allo stato e permettono di forzare il valore manualmente (come meglio dettagliato a seguire)

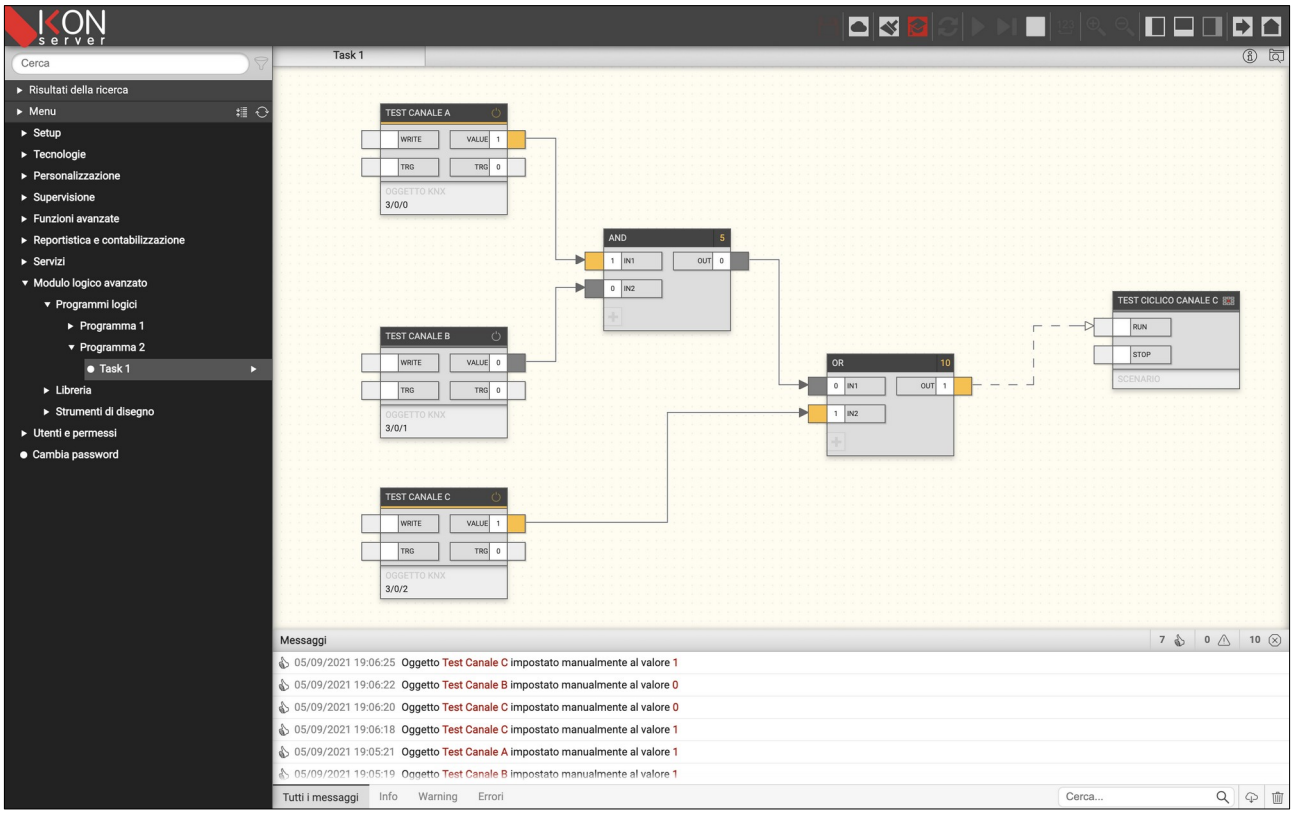
Il colore dei nodi segue la seguente convenzione:

Nodi binari	Grigio	Valore 0 (OFF)
	Giallo	Valore 1 (ON)
Nodi numerici	Blu	Qualunque valore

Durante la simulazione, l'editor riporta nell'area messaggi una serie di informazioni relative all'esecuzione dei programmi, ai cambi di stato manuali (effettuati dall'utente) e automatici (rilevati dai blocchi logici). Inoltre, durante la simulazione passo-passo, vengono riportati molti messaggi di livello “debug” che consentono un'analisi approfondita dell'esecuzione dei programmi, utile soprattutto in caso di errori o di malfunzionamenti rispetto alle intenzioni.

L'area messaggi, normalmente chiusa per offrire il massimo spazio utile alla simulazione, può essere aperta per consultare questi messaggi, il cui numero – in base alla tipologia – è sintetizzato nella parte destra della barra messaggi, visibile anche se chiusa. Per ulteriori dettagli sull'area messaggi si rimanda alla sezione 3.6.

La figura seguente mostra un esempio di simulazione con l'area messaggi aperta:



8.4 INSERIMENTO MANUALE DEI VALORI

Per impostare manualmente lo stato di un nodo, procedere nel seguente modo:

- Fare doppio click sul valore del nodo (l'etichetta diventa editabile)
- Cancellare il valore attuale ed inserire il nuovo valore
- Premere INVIO

Il colore del nodo (se digitale) cambia in funzione del nuovo valore, e questo viene passato al simulatore, che lo propaga istantaneamente (nel caso di simulazione continua) oppure al ciclo successivo di esecuzione (in modalità passo-passo).

8.5 ARRESTO DELLA SIMULAZIONE

E' possibile fermare in qualunque momento la simulazione premendo il pulsante di arresto della simulazione nella toolbar (normalmente non accessibile al di fuori della simulazione).

8.6 ESECUZIONE

Un programma può essere messo in esecuzione in due modi:

- Premendo il pulsante AVVIA nella sua scheda di dettaglio
- Premendo il pulsante ESECUZIONE nella toolbar all'interno dell'editor di uno dei suoi task

All'avvio viene generato il codice LUA corrispondente alla rete logica realizzata graficamente con l'editor; questa operazione può richiedere alcuni secondi, in funzione della complessità della logica. Al termine, un messaggio indica la fine della compilazione dello script e l'avvio dell'esecuzione.

Durante l'esecuzione, il pulsante di ESECUZIONE nella toolbar lampeggia ad indicare che l'esecuzione è attiva; in questa modalità, non è possibile aggiungere o togliere blocchi, né modificare i loro parametri. Lo stato degli oggetti viene visualizzato tramite colorazione dei nodi, ma non quello dei blocchi logici LUA (cosa che invece avviene nella simulazione).

9 STRUMENTI DI DISEGNO

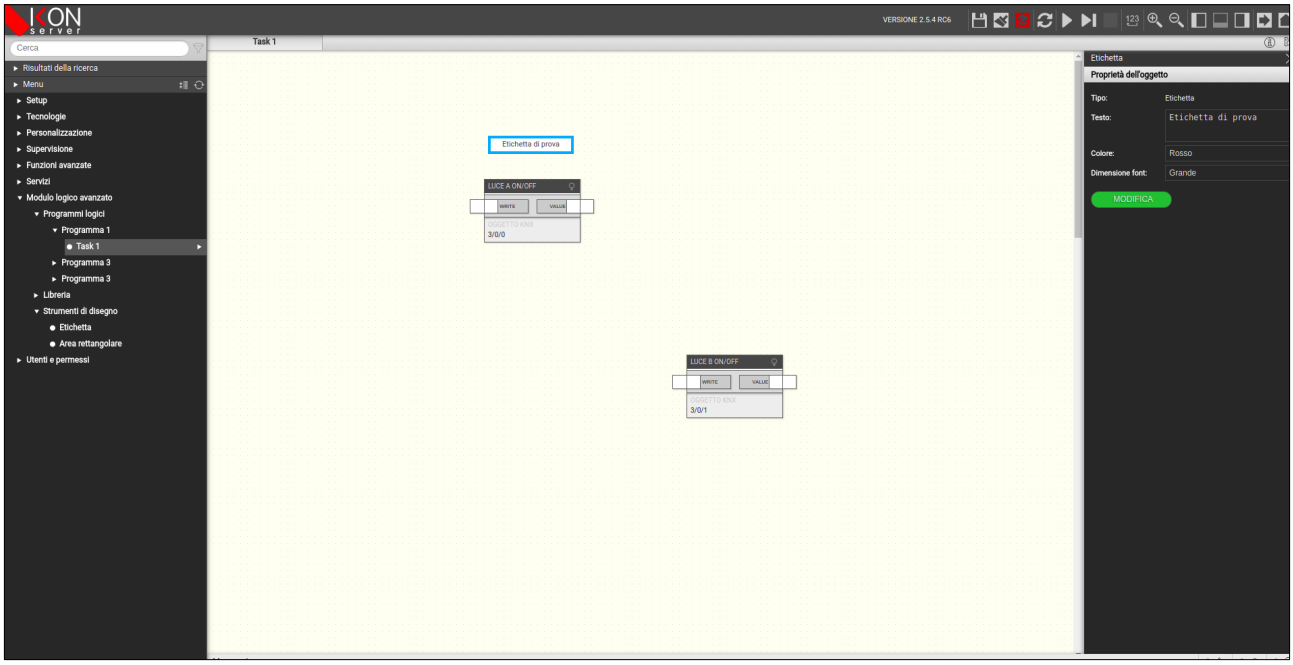
9.1 INTRODUZIONE

Al fine di aumentare la leggibilità dei programmi logici, soprattutto in caso di reti logiche complesse, l'editor mette a disposizione alcuni strumenti di disegno, con i quali l'utente può inserire annotazioni ed evidenziare aree del programma.

Questi strumenti sono disponibili nell'area “Modulo logico avanzato” del menu principale, all'interno della sezione “STRUMENTI DI DISEGNO”; essi possono essere trascinati mediante *drag&drop* nei programmi logici alla stregua delle altre tipologie di oggetti, come già visto in precedenza.

9.2 ETICHETTE

Le etichette permettono di inserire testo libero all'interno dei programmi; è possibile inserire un numero illimitato di etichette per ogni programma logico.



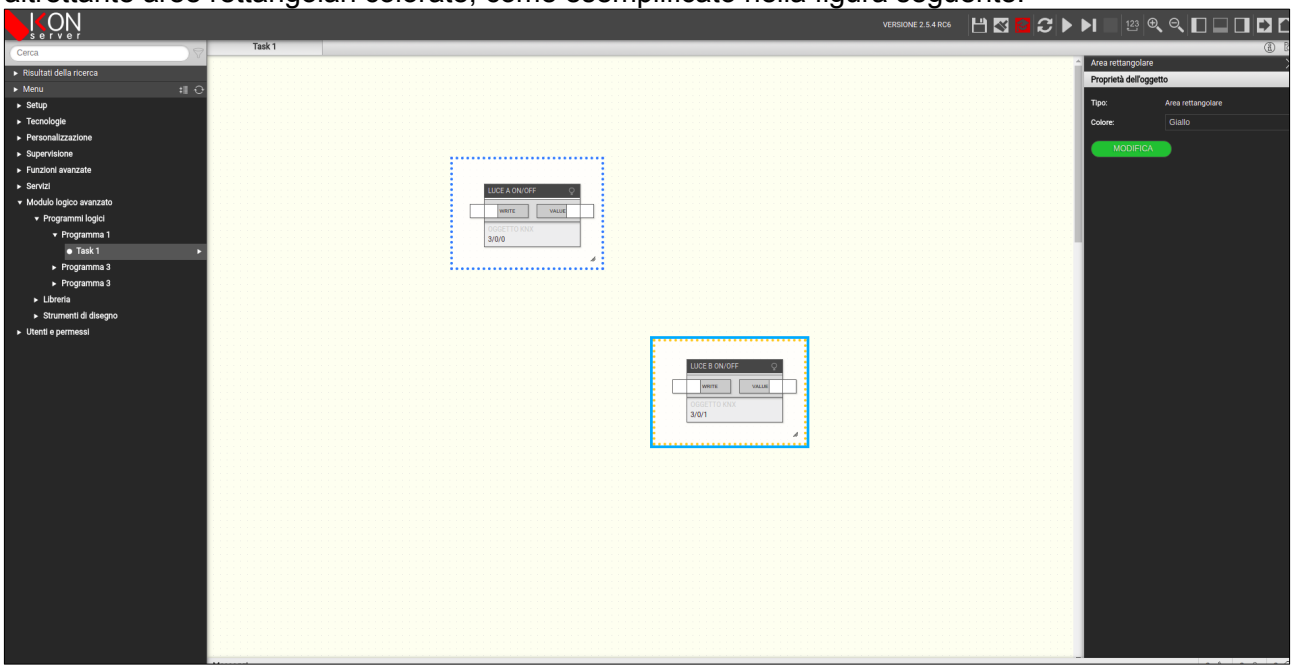
Una volta trascinata un'etichetta nel programma logico, e posizionata nel punto desiderato, è possibile personalizzarla aprendo il pannello dettagli (dopo averla selezionata); sono disponibili le seguenti opzioni:

Testo	Testo visualizzato all'interno del programma logico
Colore	Permette di scegliere il colore del testo
Dimensione font	Permette di scegliere la dimensione del carattere

Le etichette possono essere rimosse dai programmi logici mediante il corrispondente pulsante “ELIMINA” nel pannello dettagli, oppure premendo direttamente il tasto CANC della tastiera una volta selezionate.

9.3 AREE RETTANGOLARI

E' possibile evidenziare una o più porzioni del programma logico trascinando dal menu principale altrettante aree rettangolari colorate, come esemplificato nella figura seguente:



Una volta trascinata un'area rettangolare in un programma, è possibile:

- Ridimensionarla, trascinando l'apposito cursore nel suo angolo in basso a destra
- Cambiare colore del bordo, utilizzando il selettore “colore” nel pannello dettagli

Le aree rettangolari vengono sempre disegnate sotto i blocchi e le relative connessioni; esse inoltre non supportano la selezione multipla (come i blocchi o le etichette), quindi per personalizzarle o per rimuoverle dal programma è necessario fare click su di esse, una alla volta, ed utilizzare gli strumenti nel pannello dettagli (cambio colore e pulsante “ELIMINA” per rimuoverle dal programma).

10 CODICE LUA

10.1 PREMESSA

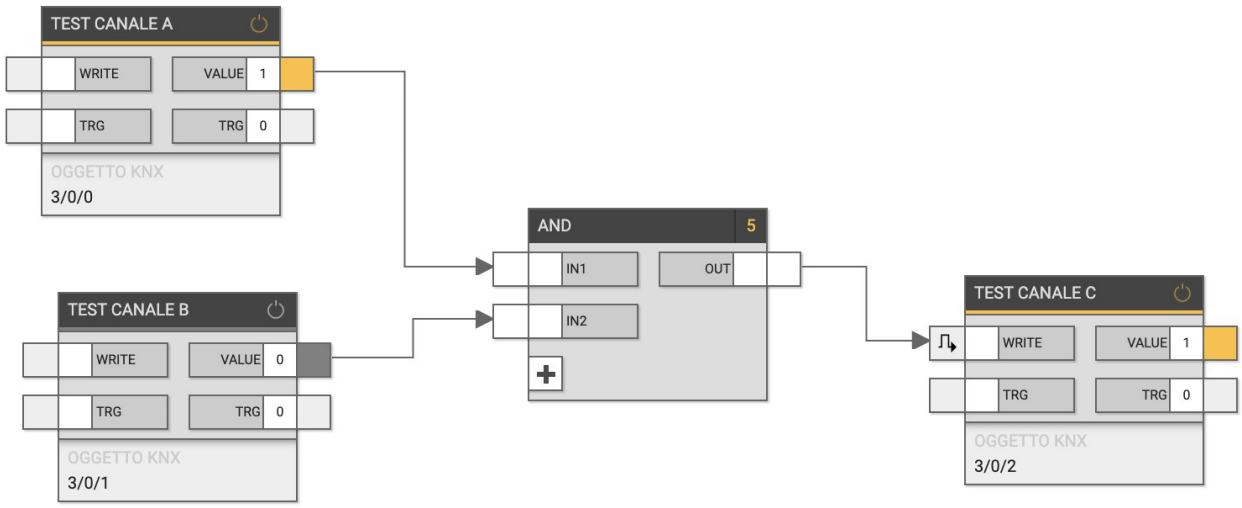
Ogni programma logico è associato ad uno script LUA generato automaticamente dall'editor grafico, quando si avvia la simulazione oppure lo si mette in esecuzione; non è prevista la modifica manuale di questi script.

Una volta generato, lo script viene messo in esecuzione attraverso un apposito processo in background; questo processo esegue ciclicamente lo script, effettuando le operazioni in esso descritte, che in questo capitolo si intende sinteticamente presentare. E' possibile visualizzare il codice associato ad un programma nella sua scheda di dettaglio, come visto in precedenza.

Il fatto che il programma logico si appoggi ad uno script generato automaticamente, costituisce un forte elemento di flessibilità, essendo di fatto un listato di codice; ovviamente, questo codice essendo prodotto da un editor grafico, è molto diverso rispetto a quello che scriverebbe un programmatore umano per ottenere lo stesso risultato.

10.2 STRUTTURA DELLO SCRIPT

Di seguito viene riportato il codice associato ad un programma costituito da un solo task, contenente una AND tra due oggetti KNX che ne comanda un terzo, esempio già visto in precedenza in questo manuale:



Questo esempio di semplice script permette più facilmente di evidenziare le sezioni in cui è articolato il codice generato dall'editor:

`-- SCRIPT 3554 GENERATED AT 2021-09-05 23:04:22`
 	
`function f_app_handler()`
`  while 1 do`
 	
`  -- INPUT`
`  v4360,t4360=getInputVal(4360)`
`  v4363,t4363=getInputVal(4363)`
`  v4366,t4366=getInputVal(4366)`
	Commento iniziale
	
	Funzione "main" con ciclo infinito
	
	Lettura dello stato degli oggetti di IKON SERVER

`-- PRE-FUNCTIONS`	Funzioni di pre-elaborazione (se presenti)
`-- FUNCTIONS`	Codice associato al task

`-- START TASK "Task 3" [4691]`
 	
`-- luafunction "andn" [4694]`
`v4695=v4360 --- Relation 10913`
`v4696=v4363 --- Relation 10914`
`v4697=dlab.andn(4694,2,v4695,v4696)`
`v4366=v4697 --- Relation 10915`
	Le relazioni assegnano il valore degli oggetti di IKON SERVER (letti all'inizio dello script ed associati ad altrettante variabili LUA) alle variabili che rappresentano i nodi di ingresso del blocco AND, e richiamano la funzione di libreria associata

`-- Swap input vars to output vars`
`o4366=v4366`
`-- END TASK`
	Le variabili destinate ad essere passate in uscita vengono inoltre assegnate a variabili di appoggio

`-- POST-FUNCTIONS`	Funzioni di post-elaborazione (se presenti)

`-- DEBUG`
`setDebugVal(4695,v4695)`
`setDebugVal(4697,v4697)`
`setDebugVal(4696,v4696)`
 	
`-- OUTPUT`
`setOutputVal(4366,o4366,false,false,0.1)`
	Richiamo delle funzioni di output per il debug (nodi interni ai blocchi di libreria) e per l'esecuzione vera e propria dei comandi

`coroutine.yield()`
`end`
`end`
	Chiusura del codice della funzione "main" con loop infinito

`-- GENERATED in 0.8304 seconds.`	Commento finale

10.3 LETTURA INGRESSI

All'inizio di ogni ciclo viene letto lo stato aggiornato di tutti gli oggetti di IKON SERVER presenti nel programma, siano essi utilizzati come ingressi della logica o come uscita; anche nel caso di oggetti solo comandati, infatti, il programma logico confronta il valore determinato dalle funzioni logiche ad ogni ciclo di esecuzione, e se questo non è coerente con lo stato attuale, viene mandato un comando.

La lettura in ingresso dei valori degli oggetti viene effettuato con queste direttive (una per ogni oggetto)…

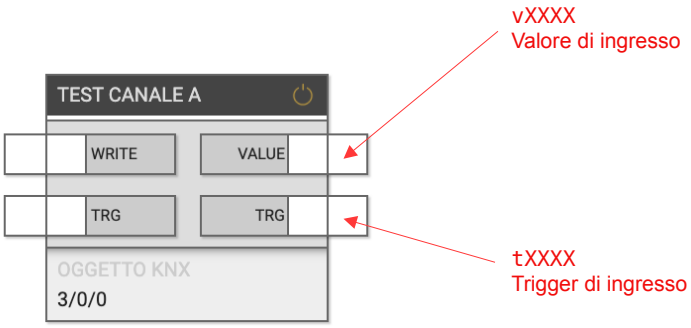
```
-- INPUT
v4360,t4360=getInputVal(4360)
v4363,t4363=getInputVal(4363)
v4366,t4366=getInputVal(4366)
```

… a cui viene passato l'identificativo univoco dell'oggetto (sia esso un oggetto semplice che un oggetto composito di IKON SERVER), e che restituiscono due valori, assegnati ad altrettante variabili LUA:

- Valore corrente: viene assegnato alla variabile vXXXX
- Trigger: viene assegnato alla variabile tXXXX

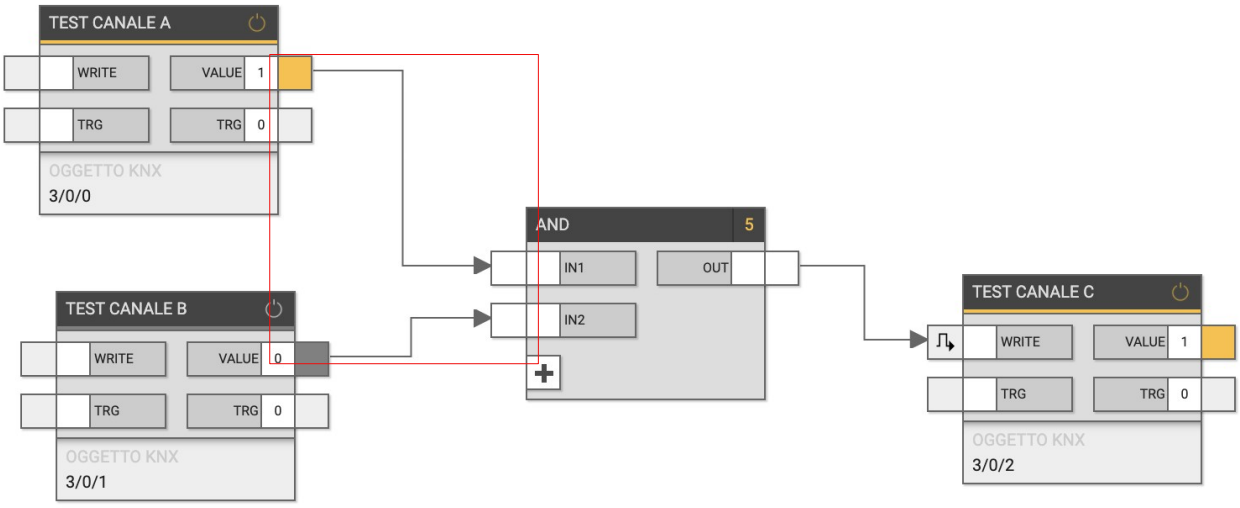
dove ovviamente XXXX rappresenta l'ID di ogni oggetto.

Il trigger viene messo a 1 se, tra la precedente lettura di stato e questo ciclo di esecuzione, è stato ricevuto un cambio di stato dal campo; al ciclo successivo, se nel frattempo non è nuovamente cambiato lo stato, il trigger viene posto a 0. Questa variabile rappresenta nel codice LUA, quello che visivamente è il nodo TRG di ingresso (lato destro) del blocco che rappresenta l'oggetto nell'editor:



10.4 CONNESSIONI

Come anticipato in precedenza, le connessioni determinano il passaggio di valore da un nodo ad un altro, e nel codice si traducono in assegnazioni di variabili LUA; ogni nodo ha un proprio ID (che fa riferimento al database di progetto) e i nomi delle variabili riflettono questo ID; con riferimento alla logica presa ad esempio, le connessioni di ingresso alla logica AND…



… si traducono nel seguente codice LUA:

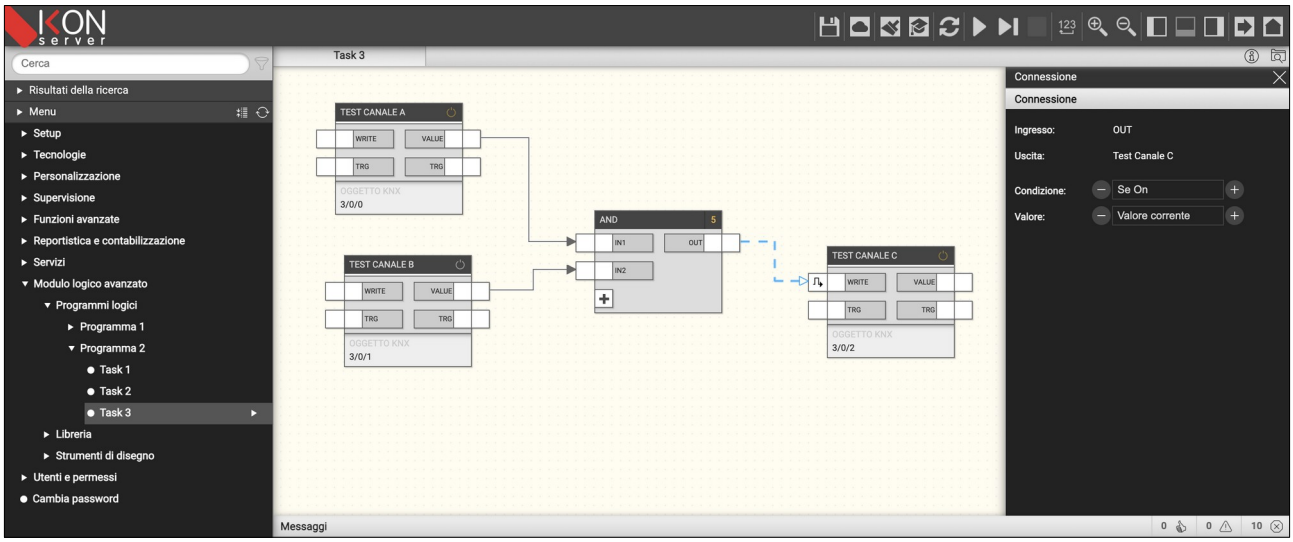
```
v4695=v4360                    --- Relation 10913
v4696=v4363                    --- Relation 10914
```

dove rispettivamente

- 4360 e 4363: rappresentano nel database gli oggetti CANALE A e B
- 4695 e 4696: rappresentano nel database i nodi di ingresso del blocco AND

Nel caso di connessioni con condizione, l'assegnazione tra variabili viene condizionata da un

controllo sul valore della prima variabile, e solo in quel caso avviene l'assegnazione: una connessione di questo tipo, pensata per passare al CANALE C il valore della AND solo se è verificata…



… si traduce nel seguente codice LUA ...

```
if v4697 == true then v4366=v4697 end          --- Relation 10915 (condition: true)
```

… al posto del precedente senza condizioni:

```
v4366=v4697                                    --- Relation 10915
```

Se si imposta anche un valore alla connessione (es: ON al posto di "valore corrente") il codice si modifica ulteriormente nel seguente modo:

```
if v4697 == true then v4366=true end           --- Relation 10915 (condition: true - value: 1)
```

10.5 BLOCCHI LOGICI

I blocchi di libreria corrispondono all'interno del codice LUA alla chiamata di una funzione, come nell'esempio precedente in questa riga di codice (per il blocco AND):

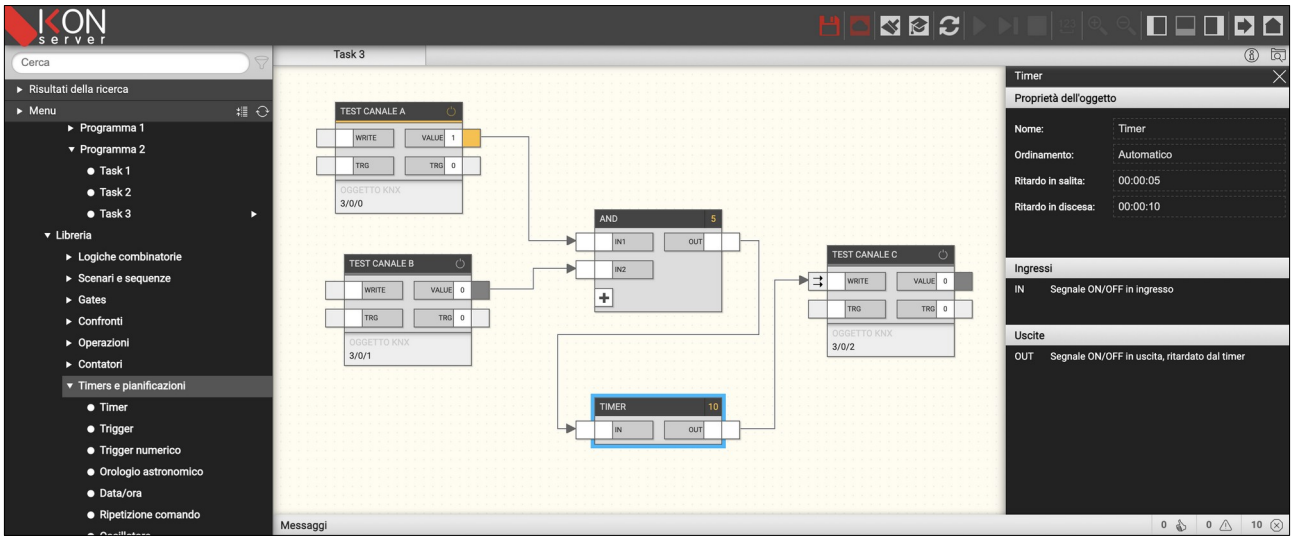
```
v4697=dlab.andn(4694,2,v4695,v4696)
```

Ogni funzione di libreria prevede un diverso numero di argomenti in ingresso; tipicamente vengono passati:

- ID del blocco nel database di IKON SERVER
- Numero di ingressi / uscite, se dinamico
- Valori per i nodi di ingresso (ovvero il valore delle corrispondenti variabili LUA, assegnato nelle righe di codice precedenti attraverso le connessioni)
- I parametri impostati nelle **opzioni** del blocco, se previste

Le funzioni di libreria restituiscono uno o più valori, a seconda dei casi, che vengono assegnati ad una o più variabili LUA, le quali rappresentano gli ID nel database con cui sono rappresentati i nodi di uscita del blocco, e tipicamente vengono a loro volta assegnate ad altre variabili LUA con le connessioni di uscita.

Alcune funzioni operano su base tempo; in questo caso, ii valori restituiti dipendono dal tempo passato dal cambio di valore degli ingressi, oppure in base all'orologio di sistema. Modificando ad esempio la logica precedente introducendo un TIMER prima del comando di CANALE C…



… il codice si modifica nel seguente modo:

```
-- SCRIPT 3554 GENERATED AT 2021-09-05 23:04:22

function f_app_handler()
     while 1 do

          -- INPUT
          v4360,t4360=getInputVal(4360)
          v4363,t4363=getInputVal(4363)
          v4366,t4366=getInputVal(4366)



          -- PRE-FUNCTIONS



          -- FUNCTIONS

          -- START TASK "Task 3" [4691]

          -- luafunction "andn" [4694]
          v4695=v4360                    --- Relation 10913
          v4696=v4363                    --- Relation 10914
          v4697=dlab.andn(4694,2,v4695,v4696)
          v4366=v4697                    --- Relation 10915


          -- luafunction "timer" [4699]
          v4700=v4697                    --- Relation 10947
          v4701=dlab.timer(4699,v4700,5,10)
          v4366=v4701                    --- Relation 10948




          -- Swap input vars to output vars
          o4366=v4366
          -- END TASK

          -- POST-FUNCTIONS



          -- DEBUG
          setDebugVal(4695,v4695)
          setDebugVal(4697,v4697)
          setDebugVal(4696,v4696)
```

Funzione timer, a cui vengono passati i parametri di ritardo in salita (5 secondi) e discesa (10 secondi). Il valore di uscita dipende da un temporizzatore interno alla funzione di libreria.

```
            -- OUTPUT
            setOutputVal(4366,o4366,false,false,0.1)


            coroutine.yield()
        end
end
-- GENERATED in 0.8304 seconds.
```

10.6 OUTPUT E DEBUG

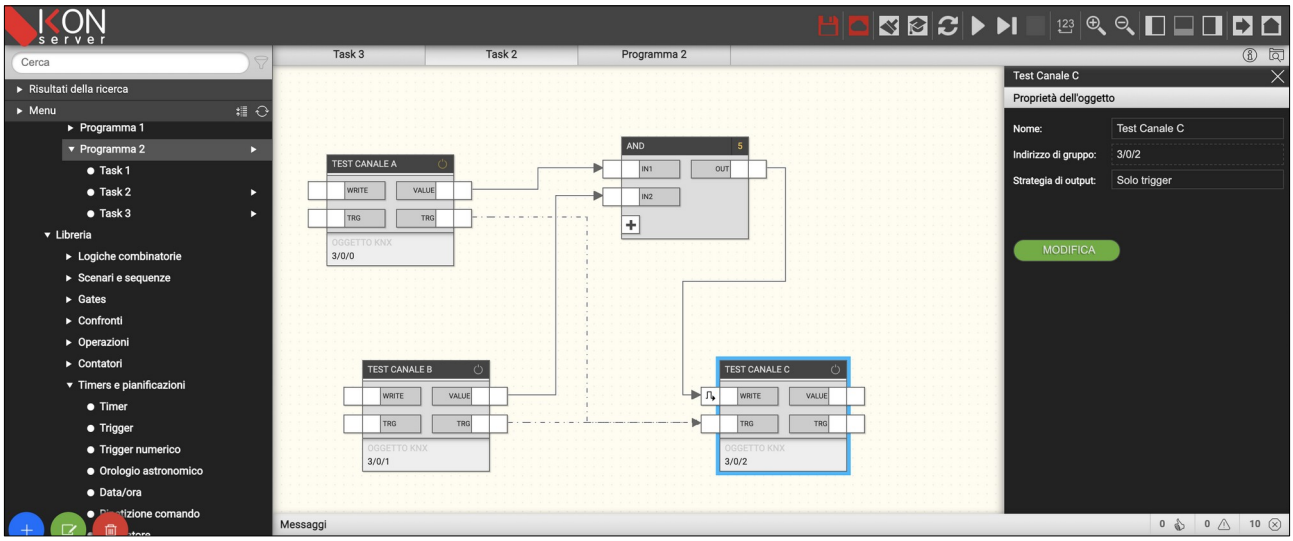
Il comando degli oggetti di uscita viene effettuato dalle chiamate alla seguente funzione:

```
setOutputVal(4366,o4366,false,true,0.1)
```

Questa funzione si aspetta i seguenti parametri:

- ID dell'oggetto da comandare
- Valore da comandare (variabile LUA assegnata alla fine della catena di assegnazioni, che rappresentano le connessioni)
- Trigger
- Strategia di output
- COV (change of value) ovvero soglia minima di variazione di valore, al di sotto della quale non viene effettuato il comando (predefinito 0.1)

Se si collega una connessione al nodo **trigger di uscita**, al posto di "false" come nella riga di codice precedente, viene passato il valore di quel trigger: nel caso di comando di CANALE C con **strategia di output "solo trigger"** e nodo TRG collegato ai trigger dei due oggetti di ingresso…



… nel codice LUA viene inserita una prima riga di codice, che imposta il trigger dell'oggetto in uscita come OR tra i trigger collegati in ingresso…

```
t4366=t4360 or t4363
```

… e il comando di output diventa il seguente:

```
setOutputVal(4366,o4366,t4366,false,0.1)
```

Come si può intuire dagli esempi, il parametro strategia di output valore “true” nel caso predefinito di cambio di valore + trigger, “false” nel caso di solo trigger.

Oltre alle righe di comando, nella parte finale dello script vengono inserite anche le direttive di debug seguenti, una per ogni nodo di ingresso o uscita di tutti i blocchi di libreria presenti nel task:

```
-- DEBUG
setDebugVal(4695,v4695)
setDebugVal(4697,v4697)
setDebugVal(4696,v4696)
```

Durante la simulazione, queste direttive passano al “motore” di IKON SERVER il valore intermedio di ogni nodo dei diversi blocchi (che come abbiamo visto in precedenza, sono identificati ciascuno con un proprio ID univoco, ed una corrispondente variabile LUA), il che permette di visualizzarne lo stato a video, colorando i nodi durante la simulazione.

Durante l’esecuzione normale, le direttive di debug vengono invece ignorate, per non rallentare l’esecuzione del codice con comandi inutili ai fini della logica.

11 APPENDICI

11.1 GLOSSARIO

Modulo logico	La nomenclatura da catalogo è “Modulo Logico”.
Programma logico	Rete logica composta da uno o più task logici.
Task logico	Rete logica composta da uno o più *blocchi* e logiche interconnesse. Ogni *unità logica* può contenere fino a 16 programmi logici.
Blocco logico	Blocco inseribile all'interno di un *programma logico* per svolgere una specifica funzione, interagendo con altri *blocchi* attraverso *nodi* di ingresso e/o uscita
Blocco	Blocco inseribile all'interno di un *programma logico* per leggere e/o scrivere informazioni sul bus domotico, interagendo con altri *blocchi* attraverso *nodi* di ingresso e/o uscita
Nodo	Singolo elemento di un blocco logico che mette a disposizione in ingresso o uscita una specifica informazione; i nodi possono essere interconnessi con altri nodi attraverso altrettante *connessioni*
Connessione	Collegamento tra due *nodi* di altrettanti *blocchi*. La connessione ha un “verso” che determina l'ordine con cui vengono scambiate le informazioni tra i *nodi*; in particolare, lo stato del nodo di *origine* viene passato al nodo di *destinazione*
Editor	Ambiente grafico di configurazione dei *programmi logici*. Permette di costruire *programmi logici* per le *unità logiche* presenti nel progetto, e di scaricare all'interno di questi ultimi le informazioni necessarie per la loro esecuzione.