UNIVERSITÀ
DEGLI STUDI
DI PADOVA

DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

CORSO DI LAUREA IN INGEGNERIA ELETTRONICA

"Descrizione di un progetto e dell'implementazione del software per il sistema di automazione di un impianto di condizionamento in una nave da crociera"

Relatore: Prof. Stefano Vitturi

Laureando: Gabriele Passarin

ANNO ACCADEMICO 2022 – 2023

Data di laurea 21/03/2023

Dedicato a Giulia e alla mia famiglia

Indice

1. Introduzione

1.1 Descrizione dell'azienda

DSG Automation viene fondata nel 1996 a Camponogara (VE). Fin da subito l'azienda si concentra nel settore dell'automazione dei processi industriali e del system integration, cioè quell'insieme di tecnologie che attuano una gestione di dati tramite software per condividere le informazioni tra vari sottosistemi. Successivamente si evolve come fornitore di soluzioni elettroniche ed informatiche per il controllo di processi produttivi industriali in molteplici settori e piattaforme di sviluppo. In questo modo consolida i rapporti con i loro clienti, i quali riconoscono l'eccellenza e la professionalità oltre al servizio di elevata affidabilità.

La difficoltà elevata delle problematiche tecniche studiate e risolte ha permesso a DSG di acquisire e maturare una capacità di "problem solving" che da allora è un pilastro dell'azienda. Questo concetto è stato impiegato nell'analisi e nel progetto di soluzioni dedicate, permettendo rapidità e affidabilità degli interventi tecnici. L'azienda è caratterizzata inoltre da una assistenza al cliente completa: dall'analisi di fattibilità, dimensionamento, fornitura dell'hardware, collaudo del sistema, messa in servizio dell'impianto e infine il supporto post-vendita.

Lo sviluppo di software per controllori programmabili ha introdotto la necessità di sviluppare applicazioni di supervisione SCADA, un settore caratterizzato dal forte grado di innovazione, per andare incontro alle richieste del cliente in modo ottimale.

Nell'ultimo decennio l'azienda si è ampliata introducendo la divisione DSG Robotics, la quale si concentra sul controllo di robot industriali Scara, antropomorfi e collaborativi; inoltre, nel 2019 è stata creata Data 4 Industry, che sviluppa applicazioni MES, IoT e di analisi dei dati nell'ambito di industria 4.0.

1.2 Obiettivi della tesi

Questo elaborato ricostruisce i passi compiuti nell'attività di tirocinio svolta in DSG Automation e fornisce una descrizione globale del progetto sviluppato con dispositivi Schneider per l'automazione dell'impianto HVAC all'interno di una nave da crociera. Purtroppo, non è stato possibile prendere parte allo sviluppo del codice in quanto al mio arrivo in azienda il progetto era ormai in fase conclusiva. Per questo motivo è parso

conveniente puntare a scrivere una tesi che descrivesse il progetto in ogni sua parte. In questo modo è stata realizzata anche una revisione del progetto.
Prima di tutto è stato esaminato il materiale fornito dal cliente contenente le specifiche di sviluppo. Di particolare importanza si evidenziano i diagrammi P&I e gli schemi elettrici dell'impianto, il documento di *CONTROL & PHILOSOPHY,* il quale ancor prima degli schemi tecnici raggruppa numerose informazioni sul dimensionamento e il funzionamento dell'impianto e infine le specifiche per il monitoraggio del consumo energetico. Questi documenti mi hanno portato per la prima volta a ricercare informazioni sui PLC Schneider Electric qui utilizzati e grazie al loro sito internet ho potuto esaminare con facilità manuali, guide e altri documenti fondamentali per capire la struttura e il funzionamento di questi dispositivi, oltre al materiale relativo agli ambienti di sviluppo Unity Pro XL e Citect SCADA. Una volta compresa la topologia della rete ho approfondito l'aspetto teorico dei protocolli utilizzati e ho studiato il funzionamento dei dispositivi per le interfacce di rete. Infine, mi sono concentrato sullo sviluppo vero e proprio del progetto, analizzando soprattutto ogni DFB Type e ogni sezione del MAST, elementi spesso condivisi tra la maggior parte dei progetti.
Finita la fase di sviluppo il cliente ha richiesto di apportare una modifica, riguardante soprattutto il software di supervisione, che permettesse di monitorare in tempo reale il consumo di energia elettrica da parte dell'impianto. In questa tesi si è voluto analizzare anche il procedimento di implementazione di questa richiesta dedicandole un capitolo a parte, volendo dare un senso di separazione temporale ma anche per evidenziare come sia possibile soddisfare abbastanza facilmente una richiesta così pervasiva partendo da un progetto già esistente.

1.3 Tecnologie HVAC

HVAC sta per Heating Ventilation and Air Conditioning, ovvero "riscaldamento, ventilazione e condizionamento dell'aria". La progettazione di sistemi HVAC si basa su tematiche tecniche conosciute e studiate da molti decenni come la fluidodinamica e la termodinamica, punti comuni della progettazione in vari ambiti come quello meccanico, civile, navale, industriale. Negli ultimi anni gli studi di questo settore hanno portato a grandi innovazioni, grazie alla richiesta di un mercato sempre più in espansione che per esempio si è spostata dall'occuparsi del condizionamento di grandi edifici a quello dei mezzi di trasporto.

2. Descrizione dei PLC e dell'ambiente di sviluppo

2.1 Architettura e funzionamento del PLC

I PLC sono dispositivi basati su microprocessore. Per via della loro resistenza alle condizioni estreme, vengono molto utilizzati nell'ambito industriale, una realtà in cui grandezze come temperatura, pressione, umidità e altri ancora possono raggiungere valori intollerabili per attrezzature più fragili.

Il PLC è uno strumento composto da vari moduli, come mostrato in figura 2.1.1, ognuno con una funzione diversa, che vengono installati su appositi *rack* ovvero supporti metallici che facilitano l'organizzazione e il cablaggio di questi dispositivi. Principalmente consistono di una o più unità centrali contenenti la CPU, un modulo di alimentazione e tutte le altre schede che permettono la comunicazione con il campo o con altri PLC.

L'unità centrale oltre a contenere la CPU, ospita anche la memoria del dispositivo ed esegue ciclicamente il programma utente che è salvato al suo interno.

In generale la memoria può essere realizzata con diverse tecnologie, i più moderni utilizzano ad esempio una memoria EEPROM, FLASH EPROM o RAM e a seconda del costruttore questa viene suddivisa fisicamente o logicamente in sezioni che realizzano compiti diversi. La memoria viene suddivisa solitamente in tre parti: memoria di sistema, necessaria per il normale funzionamento del PLC, la memoria che contiene il programma utente, infine la memoria dati contenente i valori in continuo aggiornamento delle variabili.

I tipi di schede per l'input e l'output sono di varia natura, anche in base al tipo di segnale che devono trattare. Permettono di scambiare informazioni tra il campo e l'unità centrale; prima ricevendo segnali elettrici inviati da relè, oppure leggendo valori misurati da sensori o altri dispositivi digitali o analogici, in seguito avviene da parte dell'unità centrale l'elaborazione e l'invio di una risposta ad attuatori o altri dispositivi di controllo relativa agli input ricevuti. Costituiscono una parte fondamentale del sistema.

I moduli di comunicazione permettono al PLC di interfacciarsi con tutte le reti di comunicazione più diffuse. Rendono quindi possibile il collegamento a periferiche standard ma soprattutto permettono al sistema un funzionamento distribuito nello spazio, grazie al minore utilizzo di cablaggio. Questo dettaglio è particolarmente vantaggioso in quanto permette di mantenere una migliore schermatura del segnale propagato e di utilizzare protocolli di rete ben definiti che spesso hanno meccanismi di rilevazione e correzione degli errori di trasmissione. Ad esempio, nel progetto oggetto di questa tesi vedremo come le unità

centrali utilizzate saranno fornite di ingresso ethernet, che consente di collegarle direttamente a uno switch e quindi a moduli remoti o ai computer di supervisione senza dovere utilizzare una scheda dedicata.

Il modulo di alimentazione assicura che la tensione di rete sia stabilizzata sui valori richiesti dai vari componenti. Inoltre, fa sì che i dispositivi non siano esposti a fenomeni come sovratensioni e picchi di corrente.

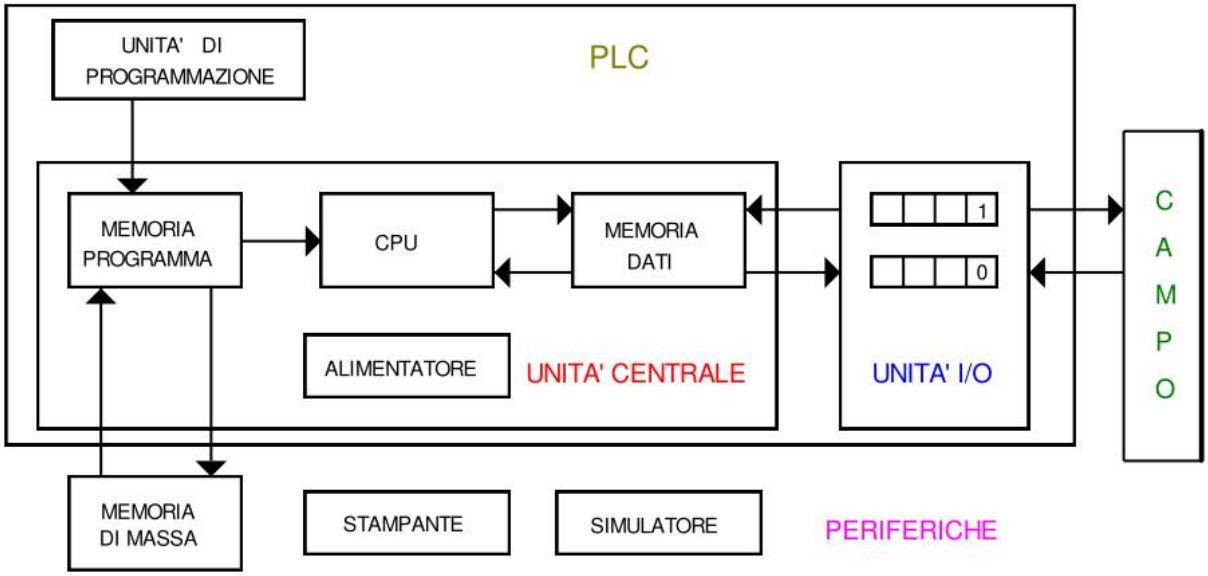


Figura 2.1.1 Architettura di un PLC in cui la memoria di sistema viene rappresentata includendola nella CPU.

Il dispositivo utilizzato come unità centrale in questa attività è il modulo BMXP3420302, della gamma Modicon M340 di Schneider Electric. Ha una capacità di gestione di 1024 I/O digitali e 256 I/O analogici, utilizza un Bus CANOpen per comunicare con fino a 63 moduli slave e in aggiunta possiede una connessione integrata con la rete Ethernet.

La memoria interna è del tipo RAM ed è suddivisa in:

256kB per i dati interni;

3584kB per costanti e simboli del programma;

Per un totale di 4096kB. In più vengono fornite due schede di memoria Flash per il backup e per la prima configurazione dei server di rete.

Gli oggetti di linguaggio (language object) sono le tipologie di dato che possono essere allocati dentro la memoria dell'unità centrale. Ogni tipo viene rappresentato nel formato "%[tipologia di dati][quantità di memoria allocata]" che viene anche detto indirizzo topologico. Le diverse tipologie di dato comprendono:

- %S o %M per i dati associati alla memoria interna;
- %K per le costanti statiche;

- %I o %IW per gli input di tipo digitale o analogico;
- %Q o %QW per gli output di tipo digitale o analogico;

Quindi si possono distinguere:

- %SW %MW %KW che indicano dati allocati in una WORD;
- %SD %MD %KD %ID %QD che indicano dati allocati in una DOUBLE-WORD;
- %SF %MF %KF %IF %QF che indicano dati allocati di tipo REAL.

Il PLC opera eseguendo il programma utente salvato nella sezione apposita di memoria in modo ciclico: inizia eseguendo la prima istruzione e prosegue in modo sequenziale fino all' ultima, quindi riparte dalla prima. Il ciclo macchina viene definito come il periodo che intercorre tra l'esecuzione della prima istruzione all'ultima. Il tempo in cui un ciclo macchina viene eseguito viene chiamato tempo di ciclo. Sebbene questo tempo possa essere calcolato analiticamente sommando i tempi delle singole istruzioni, non fornisce un parametro che caratterizza il PLC. Infatti, nell'esecuzione del ciclo, dipendentemente dai valori delle variabili, verranno eseguiti dei rami anziché altri, condizionando anche la durata del processo e impedendo di fatto al tempo di ciclo di essere costante. Come indicazione generale, nei PLC moderni il tempo di ciclo normalmente non supera i 50-100ms, sufficienti per monitorare le variazioni degli ingressi e controllare le variazioni delle uscite che abbiano periodi di pochi Hertz, come tipicamente accade con i macchinari utilizzati normalmente. In caso di necessità è sempre possibile associare dei moduli dedicati a quei processi che non rispettano tale limite.
La CPU Schneider Electric utilizzata consente di gestire un task ciclico/periodico principale, un task *fast* periodico, nessun task ausiliario e fino a 64 task su evento.

2.2 Ambiente di sviluppo Unity Pro

L'ambiente di sviluppo Unity Pro, nello specifico Unity Pro XL, è lo strumento messo a disposizione da Schneider Electric per la programmazione, compilazione e simulazione dei programmi da caricare nei PLC.
All'avvio di un nuovo progetto il primo passo consiste nella scelta del controllore da utilizzare, tra quelli compatibili. Si continua aggiungendo e configurando i moduli accessori, in modo da facilitare alla fine della fase di sviluppo la transizione dalla simulazione alla versione reale dell'impianto.
Ogni programma è formato da vari Task, il principale dei quali – il primo ad essere eseguito all'avvio – prende il nome di master Task (MAST). Si fa notare che il MAST deve obbligatoriamente essere creato ma può essere anche il solo task presente. Inoltre, al contrario degli altri task (FAST, AUX0, AUX1) che hanno una frequenza di esecuzione periodica

specificata dal programmatore, il MAST segue una esecuzione ciclica. Ogni Task è quindi diviso in *sezioni*, ovvero le *routine*, tra le quali non viene prevista una principale ma quelle che vengono create si eseguono in ordine di chiamata. Una volta terminati tutti i task, il programma ricomincia l'esecuzione dall'inizio.

Unity Pro supporta cinque linguaggi di programmazione:

- FBD (Function Block Diagram) un linguaggio grafico che descrive le funzioni come un insieme di blocchi elementari a cui vengono connessi ingressi e uscite, che a loro volta vengono connesse ai blocchi tramite semplici linee orientate da sinistra a destra. Nello stesso modo, vengono spesso collegati gli output di un blocco elementare all'input del blocco successivo, appartenente alla stessa funzione.
- SFC (Sequential Functional Chart) un altro linguaggio grafico i cui elementi principali sono gli *Step* che descrivono un insieme di azioni da compiere e le *Transizioni* che rappresentato le condizioni che quando vengono verificate fanno passare allo step successivo;
- LD (Ladder Diagram) è un linguaggio che ricorda quello usato prima dell'arrivo dei dispositivi basati su microprocessore, ovvero quello usato per descrivere graficamente il funzionamento e il progetto della logica cablata usata nei processi produttivi. È molto utilizzato tra il personale con bassa preparazione informatica;
- IL (Instruction List) è un linguaggio di basso livello simile all'assembly;
- ST (Structured Text) è un linguaggio ad alto livello basato su Pascal strutturato a blocchi. Sebbene sia meno intuitivo è un linguaggio molto potente che viene utilizzato sempre più frequentemente.

2.3 Data Editor, Derived FB Types e FB Instances

L'ambiente di sviluppo offre numerosi strumenti agli sviluppatori per organizzare il lavoro in modo efficiente.

All'avvio di un progetto viene aperto in automatico il Data Editor, uno strumento in cui appaiono in tabella tutte le variabili dichiarate, le funzioni definite dall'utente (*DFB Type*), i tipi di dato definite dall'utente DDT (*Derived Data Type*) e tutte le istanze dei Function Block. Questo strumento permette di avere sempre a portata di mano una panoramica su tutti i dati del progetto permettendo di modificarli rapidamente: selezionando una riga vuota di qualsiasi tabella è possibile dichiarare un nuovo elemento specificandone il nome, il tipo, il valore di inizializzazione ed è possibile e spesso utile aggiungere un commento che ne descriva l'impiego.

I Blocchi Funzione sono blocchi programmabili adatti a realizzare logiche di funzionamento dell'impianto realizzate dal programmatore al di fuori del task principale che viene eseguito ciclicamente dal PLC. In pratica sono un insieme di istruzioni che eseguono un particolare compito che per essere eseguite devono essere richiamate in almeno una sezione del MAST. All'interno di Unity Pro viene creata una distinzione tra Derived FB Types e Derived FB Instances. La prima è una vera e propria sezione del Data Editor e permette allo sviluppatore di creare un tipo di FB, ovvero permette di specificare parametri di ingressi e uscita, variabili private e globali e infine le routine proprie del Function Block:

- inputs: la variabile associata viene passata per valore alla variabile di istanza; eventuali cambiamenti non modificano il valore della variabile originale.
- Outputs: al contrario degli input al termine dell'esecuzione la variabile originale viene sovrascritta dal valore della variabile di istanza.
- Inputs/outputs: comprendono tutte quelle strutture o altri insiemi di variabili che comprendono al loro interno sia ingressi che uscite.

Una volta ricompilate le modifiche al progetto è possibile usare il nuovo DFB Type per creare sue varie istanze, differenziate dal nome, richiamandolo all'interno delle sezioni del task principale.
I vantaggi dei Derived Data Types e dei Derived Function Block Types sono molteplici ma i più evidenti sono quelli relativi alla portabilità del codice, sia internamente al progetto che esternamente. Per questo motivo non è possibile – oltre ad essere di fatto controintuitivo - fare riferimento alle variabili proprie del progetto ma solo a quelle definite internamente al DFB Type o ai DDT. Chiaramente questi strumenti comportano anche alcuni svantaggi, tra cui l'occupazione di memoria da parte della definizione e il fatto che per garantirne la portabilità è necessario mantenere un certo livello di generalità. Inoltre, un altro svantaggio importante è che questi modelli in certe macchine non sono modificabili finché il sistema è online.

2.4 introduzione a Citect SCADA

Citect SCADA è un software di proprietà Aveva, una multinazionale britannica specializzata in soluzioni software che ha preso in gestione questo prodotto dopo l'acquisizione da parte di Schneider Electric. In questo progetto si utilizza la versione 2016 tuttavia conviene specificare che da poco l'applicazione ha cambiato nome e ora è chiamata AVEVA Plant SCADA, introducendo un nuovo ambiente di sviluppo più rapido e intuitivo.

Citect SCADA fornisce strumenti grafici per creare le pagine che costituiscono il pannello operatore e vengono visualizzate nell'applicazione runtime SCADA. Tale applicazione permette di avviare la visualizzazione delle pagine create in modalità di sola lettura nella stazione di supervisione del cliente finale, permettendogli di supervisionare l'impianto da un'unica posizione. Oltre a numerosi oggetti forniti di default, è possibile utilizzare widget personalizzati e controlli sviluppati da terze parti espandendo notevolmente le capacità del programma. Anche forme elementari come cerchi o rettangoli, possono essere animate tramite alcune funzioni apposite in modo da essere trasformati in oggetti che seguono un certo comportamento.

Tutte le variabili create durante la fase di sviluppo vengono raggruppate in *Equipment* ovvero insiemi di variabili che individuano una struttura fisica o logica, che in modo analogo alle variabili singole, vengono salvati all' interno di un database che ne ricordi il nome, il tipo e il prefisso, come si può vedere in figura 2.4.1. Si fa notare che nel caso il progetto dello SCADA coinvolga più progetti Unity Pro – e quindi in generale più PLC - il sistema differenzia le variabili anche grazie al campo *I/O device* che individua la macchina da cui proviene la variabile. Il software è arricchito di uno strumento per l'analisi avanzata in tempo reale di trend e allarmi basato su un potente ActiveX Process Analyst configurabile in runtime, e di un'ampia connettività con applicazioni di terze parti.

Infine, il software mette a disposizione un editor di codice, che crea in automatico uno script che traduce i simboli grafici in linee di codice di alto livello con lo scopo di potere analizzare nel dettaglio il significato di ognuno di essi.

Model

Row	Name	Cluster Name	Type	Comment	Tag Prefix	I/O Device	Project
3999	AP0911.AP09111TT2L	H6244	Soglia	TEMPERATURE LOW THRESHOLD	AP0911_AP09111TT2L	AP0911	SCH_H6244_20220815
4000	AP0911.AP09112TT2L	H6244	Soglia	TEMPERATURE LOW THRESHOLD	AP0911_AP09112TT2L	AP0911	SCH_H6244_20220815
4001	AP0911.AP09113TT2L	H6244	Soglia	TEMPERATURE LOW THRESHOLD	AP0911_AP09113TT2L	AP0911	SCH_H6244_20220815
4002	AP0911.AP09114TT2L	H6244	Soglia	TEMPERATURE LOW THRESHOLD	AP0911_AP09114TT2L	AP0911	SCH_H6244_20220815
4003	AP0911.AP0911PTL	H6244	Soglia	PRESSURE LOW THRESHOLD	AP0911_AP0911PTL	AP0911	SCH_H6244_20220815
4004	AP0911.AP0911TT1TL	H6244	Soglia	ANTIFROST THRESHOLD	AP0911_AP0911TT1TL	AP0911	SCH_H6244_20220815
4005	AP0911.ESD	H6244	Ingresso	ESD STOP VENTILATION	AP0911_ESD	AP0911	SCH_H6244_20220815
4006	AP0911.PF10101.PS	H6244	Sensore	PRESSURE SWITCH PF	AP0911_PF10101_PS	AP0911	SCH_H6244_20220815
4007	AP0911.PF10101.TTIN	H6244	Ingresso_Ana	TEMPERATURE TRANSMITTER	AP0911_PF10101_TTIN	AP0911	SCH_H6244_20220815
4008	AP0911.PF10101.TTOUT	H6244	Ingresso_Ana	TEMPERATURE TRANSMITTER	AP0911_PF10101_TTOUT	AP0911	SCH_H6244_20220815
4009	AP0911.PF10101.TH	H6244	Ingresso_Ana	HUMIDITY TRANSMITTER	AP0911_PF10101_TH	AP0911	SCH_H6244_20220815
4010	AP0911.PF10101.CC	H6244	PID	TEMPERATURE PID CONTROL	AP0911_PF10101_CC	AP0911	SCH_H6244_20220815
4011	AP0911.PF10101.RH	H6244	PID	TEMPERATURE PID CONTROL	AP0911_PF10101_RH	AP0911	SCH_H6244_20220815

Figura 2.4.1 Estratto del database Equipment

3. Descrizione generale dell'impianto

3.1 Topologia dell'impianto e dimensionamento

Il progetto si occupa della realizzazione del sistema di condizionamento dell'aria all'interno di una nave da crociera. Complessivamente vengono utilizzati 68 PLC: uno per ogni unità di trattamento dell'aria (67 AHU, Air Handling Unit) e uno per il sistema di gestione della sicurezza SMCS. La topologia della rete vede i PLC collegati agli AHU come slave che vengono collegati tramite rete Ethernet al software di supervisione SCADA posizionato nella sala motori (Engine Control Room ECR) e al sistema indipendente SMCS.

Ogni AHU è riconosciuto attraverso l'utilizzo di un codice in base alla tipologia di utenza che andrà a servire, il ponte della nave, la zona verticale (Main Vertical Zone, MVZ) di riferimento e infine il numero di unità, in caso di due unità uguali.

I codici che individuano le tipologie di spazio servito dall'impianto di condizionamento sono:

AC: unità Cabina Passeggeri;

AG: unità Cambusa;

AP: unità Spazio Pubblico;

AT: unità Scale;

AS: unità Spazio di Servizio;

AW: unità Cabina Equipaggio.

Un esempio di codice potrà quindi essere: AC0421 - che starà ad indicare una AHU che servirà un insieme di cabine passeggeri situate sul ponte numero 04 della MVZ numero 2. L'ultima cifra serve a distinguere tra le zone che appartengono alla stessa categoria e zona verticale; si inizia a numerarle in ordine di apparizione dal ponte più basso.

In figura 3.1.1 è rappresentata una lista contenente tutti gli spazi della nave oggetto di questo elaborato serviti dall'impianto di condizionamento.

Le *Main Vertical Zone* - come i più conosciuti *ponti* – sono un concetto di suddivisione spaziale proprio dell'ambito navale. La legge predispone per i vascelli con una lunghezza superiore ad un certo valore una obbligata suddivisione anche per sezioni verticali. Questo argomento è legato a motivi di sicurezza ed è stato introdotto per cercare di limitare il propagarsi di incendi o allagamenti, limitando all'essenziale le aree comunicanti tra zone adiacenti. Per questo motivo, anche ridurre i cablaggi relativi all'impianto HVAC rappresenta una prova ingegneristica di particolare importanza.

n°	MVZ	ITEM	SERVED AREA	AHU CATEGORIES													ENERGY SAVING		MOTOR TYPE
				Pax & Off. cabins	PO & Crew cabins	Pub. Space >100p	Pub. Space < 100p	Pub. Sp. 100% fresh air	Pub. Sp. 100% fresh air	Pub. Sp. with recirculation	Atrium -smoke exh.	Stairwais	Galleys & Laundry	Service areas	Hospital area	Technical spaces	CO2	PROGRAMMABLE TIMER	
				1	2	3A	3B	3C	3D	3E	4	5	6	7	8	9			
1		AP.01.1.1	Dk 2 - Lower dining room			X											YES	YES	FC
2		AS.01.1.1	Dk A - Converter room ps													X	NO	-	1S
3		AS.01.1.2	Dk A - Converter room stb	X												X	NO	-	1S
4		AS.01.1.3	Dk B-A - Provisions											X			NO	-	1S
5		AW.01.1.1	DK 1 - Crew cabin aft		X												NO	-	FC
6	1	AG.08.1.1	Dk B - Bakery										X				NO	YES	1S
7		AP.04.1.1	Dk 3 - Upper dining room			X											YES	YES	FC
8		AC.04.1.1	Dk 1-8 - Pax cabins	X													NO	-	1S
9		AT.04.1.1	Dk B-9 - Stairs									X					NO	YES	FC
10		AS.05.1.1	Dk 2-3 - Scullery and Various space											X			NO	-	1S
11		AP.09.1.1	Dk 9-10 - Tamarind			X											YES	YES	FC
12		SS.01.1.1	Dk B-A - Freezing Room											X			NO	-	1S
13		AS.0B.2.1	Dk B - Main Switchboard													X	NO	-	1S
14		AS.0B.2.2	Dk B - Main Switchboard													X	NO	-	1S
15		AS.01.2.1	Dk B-A - Services spaces											X			NO	-	FC
16		SS.01.2.5	Dk B - Freezing Room											X			NO	-	1S
17		AG.03.2.1	Dk 2 - Main galley										X				NO	YES	FC
18		AG.03.2.3	Dk 2 - Main galley										X				NO	YES	FC
19	2	AP.01.2.1	Dk 2-3 - Public spaces			X											YES	YES	FC
20		AG.01.2.3	Dk 3 - Hot Galley										X				NO	YES	FC
21		AC.06.2.1	Dk 1-8 - Pax cabins	X													NO	-	1S
22		AG.08.2.1	Dk 9 - Lido galley										X				NO	YES	FC
23		AP.10.2.1	Dk 9-11 - Lido restaurant			X											YES	YES	FC
24		AG.10.2.2	Dk 10 - Tamarind Galley										X				YES	YES	FC
25		AT.11.2.1	Dk B-12 - Stairs									X					NO	YES	FC
26		AP.01.3.2	Dk 2 - Britannia restaurant			X											YES	YES	FC
27		AW.01.3.1	Dk C-A - Crew cabins aft		X												NO	-	FC
28		AG.03.3.1	Dk 2-3 - Hot Galley										X				NO	YES	FC
29		AP.04.3.1	Dk 3 - Atrium								X						NO	YES	FC
30		AP.05.3.1	Dk 2 - Atrium								X						NO	YES	FC
31		AP.06.3.1	Dk 1 - Atrium								X						YES	YES	FC
32		AC.05.3.1	Dk A-5 - Pax cabins	X													NO	-	1S
33		AC.08.3.1	Dk 6-8 - Pax cabins	X													NO	-	1S
34	3	AP.11.3.1	Dk 9-10 - Lido restaurants			X											YES	YES	FC
35		AS.0A.3.1	Dk A - Emergency Controll Room ECR													X	NO	-	1S
36		AS.0A.3.2	Dk A - Emergency Controll Room ECR													X	NO	-	1S
37		AS.0B.3.1	Dk B - Switchboard fwd													X	NO	-	1S
38		AS.0B.3.2	Dk B - Switchboard fwd													X	NO	-	1S
39		AW.0B.3.1	Dk C-A - Crew cabins fwd		X												NO	-	FC
40		AG.11.3.1	Dk 10 - Queen princess Galleys										X				NO	YES	FC
41		AT.12.3.1	Dk C-11 - Stairs									X					NO	YES	FC
42		AP.12.3.1	Dk C-11 - Public Spaces				X										YES	YES	FC
43		AG.01.4.1	Dk A - Crew galley										X				NO	YES	FC
44		AP.01.4.1	Dk A - P.O., Off. and Crew mess		X												NO	YES	FC
45		AW.01.4.1	Dk C-A - Crew cabins fwd		X												NO	-	FC
46		AC.02.4.1	Dk A-1 - Off and Pax cabins	X													NO	-	1S
47		AW.02.4.1	Dk B-A - Crew cabins		X												NO	-	FC
48		AP.04.4.1	Dk 2-3 Shop - Ball room			X											YES	YES	FC
49		AC.05.4.1	Dk 4÷8 Pax cabins	X													NO	-	1S
50		AP.09.4.1	Dk 9-10 Magrodome						X								NO	YES	FC
51		AS.11.4.1	Dk 2-10 Service space											X			NO	-	FC
52		AT.11.4.1	Dk C Stairs									X					NO	YES	FC
53		AG.12.4.1	Dk 10 Galley										X				NO	YES	FC
54		AP.01.5.1	Dk 1-2 - Public Space			X											YES	YES	FC
55		AS.01.5.1	Dk A - Medical center												X		NO	-	FC
56		AW.01.5.1	Dk C-A - Crew cabins		X												NO	-	FC
57		AL.02.5.1	Dk C - Main Laundry										X				NO	YES	FC
58		AP.02.5.1	Dk 2 - Spa & Beauty saloon					X									NO	YES	FC
59	5	AC.04.5.1	Dk A-6 - Officer & pax cabins	X													NO	-	1S
60		AP.04.5.1	Dk 3 - Casinò								X						NO	YES	FC
61		AG.05.5.1	Dk 2 - Galley										X				NO	YES	FC
62		AC.06.5.1	Dk 7-14 - Pax cabins	X													NO	-	1S
63		AP.11.5.1	Dk 9 - Public Spaces					X									NO	YES	FC
64		AT.14.5.1	Dk C-15 - Stairs									X					NO	YES	FC
58		AW.01.6.1	Dk B-1 - Crew cabins		X												NO	-	FC
59		AP.01.6.1	Dk 1-2 - Spa Area					X									NO	YES	FC
60		AS.03.6.1	Dk A - Crew Laundrette											X			NO	-	1S
61		AP.04.6.1	Dk 1-2-3 - Main show lounge			X											YES	YES	FC
62		AP.04.6.2	Dk 2-3 - Main show lounge			X											YES	YES	FC
64	6	AS.05.6.1	Dk C-3 - Service space											X			NO	-	FC
65		AC.06.6.1	Dk A-6 - Pax cabins	X													NO	-	1S
66		AS.06.6.1	Dk 8 - Wheelhouse & Capitan cabin											X			NO	-	1S
68		AC.10.6.1	Dk 7-11 - Pax cabins	X													NO	-	1S
69		AP.14.6.1	Dk 12 - Crows nest				X										NO	YES	FC
70		AT.03.6.1	Dk B-8 - FWD Stairs									X					NO	YES	FC
71		AT.14.6.1	Dk C-14 - Stairs									X					NO	YES	FC
72	7	AP.04.7.1	Dk A-5 - Off. bar & crew meeting room				X										NO	YES	FC

Figura 3.1.1 Tabella di classificazione degli spazi della nave

L’impianto è dimensionato per operare in condizioni di temperatura e umidità che siano contenute all’interno di un range. Inoltre, si devono considerare anche le condizioni atmosferiche esterne che vengono divise in *Estate* e *Inverno*.
Quando è selezionato lo stato Estate si tiene conto che all’esterno vi siano +35° C e un tasso di umidità dell’85%. In questo scenario l’impianto viene programmato per assicurare una temperatura interna di +23° C e un tasso di umidità tra il 50% e il 60%.
Quando è selezionato lo stato Inverno viene considerata una temperatura esterna di -5° C e si vuole assicurare negli spazi interni una temperatura di +22° C.
Nel progetto viene specificata anche la modalità di funzionamento in fase di Safe Return to Port (SRtP), una situazione di emergenza in cui i passeggeri vengono fatti confluire all’interno di una zona sicura nella quale deve poter essere assicurata l’accessibilità ai servizi di base. Tra questi è richiesto anche il funzionamento del sistema HVAC con le seguenti specifiche: in Estate +30° C mentre in Inverno +10°C.

3.2 SCADA e reti accessorie

Il sistema di supervisione SCADA è un software con la funzione di monitorare e controllare automaticamente i dispositivi PLC ad esso collegati. Tramite interfaccia grafica permette ad un operatore di mostrare su schermo lo stato di ogni dispositivo monitorabile e/o controllabile collegato all’impianto. Questa supervisione sull’impianto suggerisce una certa affinità con tutta una serie di dispositivi e funzioni relative alla sicurezza e alla propagazione di allarmi. Infatti, lo SCADA viene collegato via Ethernet al sistema di gestione degli allarmi IAMCS (Integrated Alarm, Monitoring and Control System), il quale è in realtà compreso nel sistema di gestione della sicurezza SMCS (Safety Management Control System) e ai dispositivi di spegnimento di emergenza ESD (Emergency Shutdown System) che sono compresi all’interno dei PLC delle AHU. È interessante notare che questi sistemi, pur essendo collegati anche ai vari PLC, comunicano spesso con essi attraverso lo SCADA, dimostrando l’importanza di comunicare il proprio stato il più rapidamente possibile.

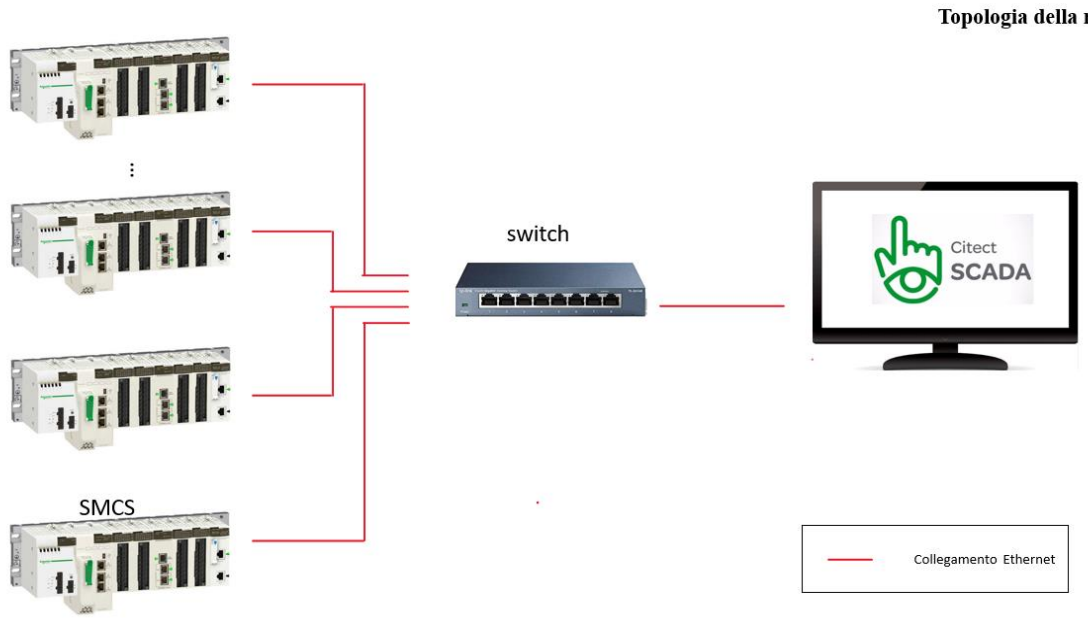


Figura 3.2.1 Topologia della rete

Le funzioni che il sistema di supervisione SCADA implementa comprendono:

- Controllo remoto di ogni AHU;
- Monitoraggio in tempo reale di ogni fan-coil (uno di questi è presente in ogni cabina passeggeri) e dei parametri ad essi associati;
- Possibilità di impostare la temperatura in ogni spazio pubblico tramite timer;
- Controllo e monitoraggio remoto di ogni FCU (Flow Control Unit) negli spazi pubblici;
- Monitoraggio dei sistemi di alimentazione degli FCU di emergenza;
- Controllo remoto e invio a schermo dello stato di ogni tipo di damper (Fire Damper FD, Smoke Damper SD, ShutOff Damper SH) e delle loro bocchette di ingresso e uscita;
- Log di dati come: temperatura spazi pubblici, temperatura degli AHU, temperatura e umidità ambientale, temperatura dell'aria proveniente dalla zona delle cabine equipaggio, temperatura di ogni stazione elettrica.

Il sistema di supervisione deve anche connettersi a un serie di reti e dispositivi atti a monitorare e assicurare la sicurezza a bordo della nave, ne sono un esempio l'ESD e il SMCS. Il Sistema di spegnimento di Emergenza (Emergency Shutdown System, ESD) è un sistema di controllo utilizzato per la gestione di sequenze di avviamento e shutdown dell'impianto. Si attiva nel momento in cui entra in azione il sistema antincendio e genera segnali che sono

passati tramite collegamento seriale al software di supervisione principale, il quale vi si interfaccia in modo da permettere all'ESD di:

- Fermare ogni tipo di apparecchiatura controllata, sia singolarmente che in gruppo;
- Monitorare lo stato di ventole e damper;
- Propagare l'allarme di discrepanza;
- Quando attivato deve poter propagare gli allarmi generati allo SCADA e gli equipaggiamenti fermati non possono tornare in funzione fino a quando il comando non viene dato dalla stazione principale;
- Inviare le azioni predisposte dal sistema SMCS senza che queste vengano sovrascritte da comandi trasmessi per il normale controllo di sistema;
- Il controllo di tutti i damper, sebbene il comando provenga dal sistema di automazione. Inoltre, in caso di conflitto, l'ESD ha la priorità sulla loro chiusura.

Il Safety Management Control System (SMCS) è un sistema progettato per la gestione di crisi su navi commerciali e mercantili che supporta gli addetti a gestire le varie fasi dell'emergenza, dalla propagazione dell'allarme alla risoluzione.

Il network è gestito da un PLC di sicurezza indipendente che viene collegato al software di supervisione in modo da:

- inviare lo stato di allarme tramite comunicazione seriale;
- ricevere i report di completamento o fallimento della messa in atto della smoke strategy.

La smoke control strategy (strategia per il controllo del fumo) è una procedura che attiva una serie di comportamenti dell'impianto HVAC da mettere in atto in caso di incendio, con lo scopo di contenere il più possibile l'emergenza e di circoscriverla nella più piccola area possibile. Nello specifico di questo progetto viene implementata una strategia che protegga dal propagarsi del fumo nelle seguenti aree:

- Vie di fuga, scale e corridoi;
- Stazioni di raduno;
- Atrio;

grazie al controllo dei damper.

3.3 Utilizzo di Modbus e CANopen

Ad ogni PLC, a seconda della zona servita, vengono collegate ulteriori reti di comunicazione che realizzano la comunicazione seriale con un numero elevato di macchine che sono collocate fino a qualche centinaia di metri dall'unità centrale. Per questioni di sicurezza ed

economiche si tende a realizzare queste reti in modo da ridurre il cablaggio e per ottenere tale risultato viene solitamente utilizzata una modalità seriale di interconnessione chiamata *daisy chain* (vedi figura 3.3.1) in cui ogni nodo intermedio della rete presenta un cavo entrante e uno uscente, tali per cui ogni dispositivo coinvolto possa comunicare con tutti gli altri. Nello specifico vedremo la rete CFCCS che comunica con Modbus e la rete "Public Fan Coil" che utilizza il protocollo CANopen. I dati raccolti da queste reti verranno infine trasmessi al software di supervisione in modo da monitorare in real-time il comportamento di un grande numero di dispositivi in modo semplice.

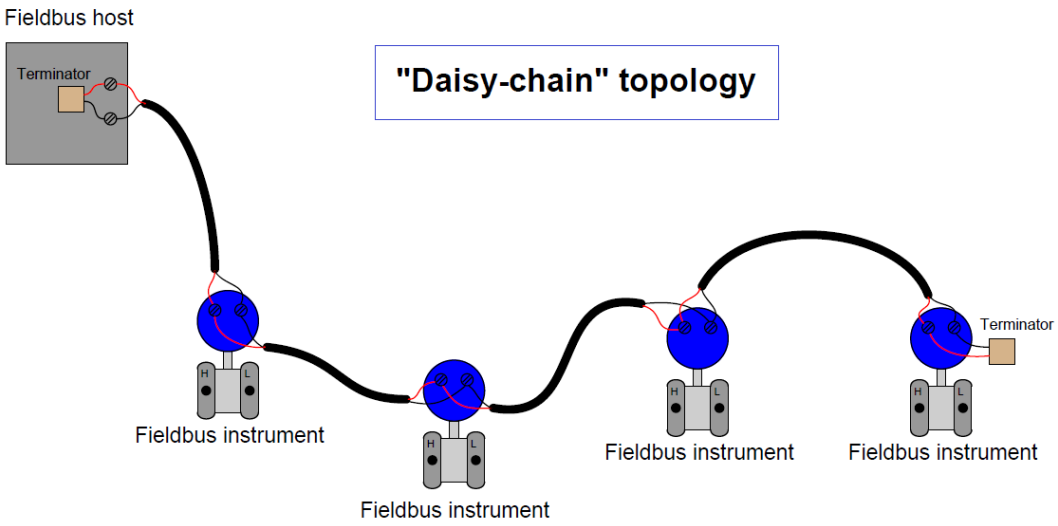


Figura 3.3.1 Esempio di topologia "Daisy Chain"

La rete CFCCS (*Cabin Fan Coil Control System*) è un sistema di comunicazione che mette in comunicazione i fan-coil di tutte le cabine passeggeri gestite da una singola AHU. Consiste nel collegare in modo seriale tutti i fan-coil ad un gateway ethernet che a sua volta trasmette i dati al PLC dell'unità di trattamento dell'aria di riferimento.

Il gateway ethernet utilizzato in questo progetto è il PowerLogic EGX100 il quale serve come accoppiatore Ethernet fra dispositivi PowerLogic System e gli altri dispositivi comunicanti che utilizzano il protocollo Modbus, inoltre offre un completo accesso a tutte le misure e gli stati dei dispositivi connessi. Questo dispositivo funge da Modbus-client della rete, in quanto è colui che richiede (e talvolta inoltra) le informazioni ai fan-coil collegati che hanno il ruolo di Modbus-server. Il protocollo permette di avere fino a 247 indirizzi unici di tipo server e per comunicare utilizza un semplice frame chiamato *Protocol Data Unit* (PDU), indipendente dagli strati di comunicazione inferiori, contenente il codice di funzione e i dati da comunicare al master. Eventualmente viene integrato un campo che indica uno specifico indirizzo e un campo dedicato al controllo di errori; in questo caso il frame prende il nome di *Application Data Unit* (ADU). Quando il Client vuole interrogare uno dei Server deve inviare un ADU contenente le seguenti informazioni:

- Il primo byte consiste nell'indirizzo del Server da raggiungere, ogni Server riceve il messaggio ma solo il vero destinatario non lo ignora;
- Il secondo byte specifica il *function code* ovvero il tipo di azione che il server deve intraprendere. È possibile definire azioni multiple utilizzando sub-function codes;
- Il terzo campo del messaggio è formato da un numero arbitrario di byte ed è denominato *Data,* contiene eventuali ulteriori informazioni sulle azioni da intraprendere;
- L'ultimo campo è utilizzato come controllo per la presenza di errori.

Application Data Unit (ADU)			
Server ID	Function code	DATA	Error check
1 byte	1 byte	N byte	1 byte

Figura 3.3.2 Suddivisione del protocollo Modbus

La risposta del server consiste in un semplice *echo* del function code con eventuali dati da comunicare nel campo dati, ovvero un PDU. In caso di errori (*exceptions*) viene settato a 1 il bit più significativo del campo function code. Come evidenziato in figura 3.3.3, si nota che non è necessario specificare l'indirizzo del destinatario in quanto l'unico interlocutore di un server può essere lo stesso client che è unico in una rete Modbus.

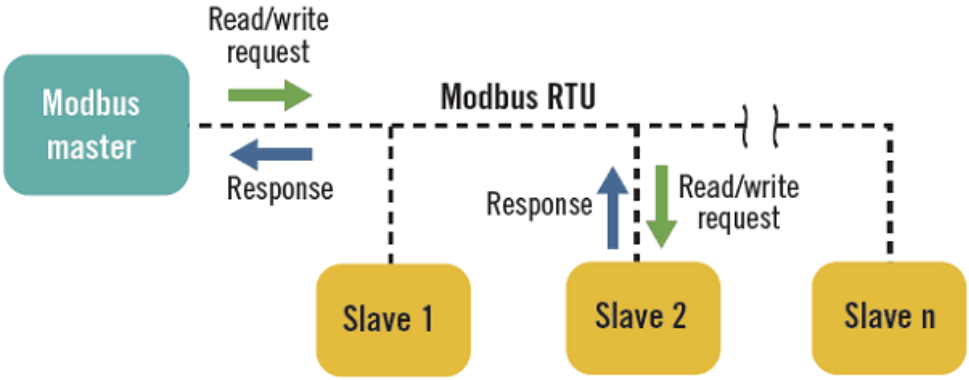


Figura 3.3.3 Comunicazione in una rete Modbus

Il modello di comunicazione seriale trova un'altra importante applicazione nella rete di comunicazione CAN (*Controller Area Network*) e il suo diretto discendente – CANopen – che è anche il tipo di bus di comunicazione che viene utilizzato sia internamente al PLC per comunicare con i propri moduli, sia per la comunicazione con una serie di reti accessorie. CANopen è un sistema di comunicazione basato su CAN che implementa i livelli dal terzo e superiori del modello OSI. Il motivo principale per cui viene usata questa rete è il fatto che

permette al programmatore di evitare di doversi occupare della comunicazione di basso livello che viene invece gestita dalla rete CAN.

Il sistema comunica attraverso lo scambio di messaggi che vengono chiamati COB (communication objects), i quali possono trasportare fino a 8 byte di dati e sono identificati da un COB-ID, elemento che ne determina la priorità, nel caso siano presenti sulla rete più messaggi. CANopen fornisce un certo numero di COB standard, per la gestione di processi critici o per il network management area (configurazione, inizializzazione e gestione degli errori) o altri ancora.

Il livello applicazione è gestito da un profilo di dispositivo standardizzato. Ciò permette ai costruttori di hardware di fornire i loro prodotti di un'interfaccia standard, rendendoli "plug and play" e quindi molto attraenti per gli sviluppatori di software che vedono estendersi la loro capacità di riutilizzare il codice, conservando la possibilità di implementare anche funzionalità specifiche.

I COB forniti da CANopen permettono di implementare all'interno di un dispositivo il network behavior desiderato, in modo tale che possano comunicare dati, segnalare errori o influenzare il funzionamento della rete.

Il dispositivo CANopen è suddiviso in tre parti logiche:

- protocol stack, gestisce la comunicazione via rete CAN;
- application software, fornisce la funzionalità per il controllo interno ma anche si interfaccia con le interfacce di hardware di processo;
- dizionario degli oggetti, un indice a 16 bit - rappresentati in forma esadecimale - che contiene una lista di tutti i tipi di dato utilizzato e conserva tutti i parametri utilizzati per la comunicazione e dall'applicazione.

Il protocollo CANopen comprende:

- SDO (Service Data Object) protocol, è utilizzato per lo scambio di dati di servizio a bassa priorità. Consiste nell'accesso alle voci contenute nel dizionario oggetti di un dispositivo da parte di un altro dispositivo. Si divide in SDO di scrittura e SDO di lettura e si attua stabilendo una comunicazione peer-to-peer di tipo Client-Server, in cui il richiedente è il client e il dispositivo proprietario del dizionario il server;
- PDO (Process Data Object) protocol, è utilizzato per i broadcast di dati di processo ad alta priorità. Il protocollo è formato da un singolo frame di 8 byte di soli dati. Un device che implementa questo protocollo ha bisogno di contenere nel proprio dizionario oggetti alcuni parametri per determinare l'identificatore COB-ID usato dal dispositivo (communication parameter record) e per indicare quali informazioni del dizionario locale trasmettere e dove memorizzare quelle ricevute (mapping record)

- NMT (Network Management), consiste in una macchina a stati supportata da tutti i dispositivi e ne determina le modalità di comunicazione. All'accensione o al reset la macchina si posiziona nello stato di *Initialization* e dopo una fase di set-up comunica l'entrata nello stato di *Pre-Operational* in cui può inviare solo SDO. Per poter inviare anche PDO la macchina passa nello stato *Operational.* Se infine la macchina viene impostata nello stato *Stopped* il dispositivo reagirà solo a comandi provenienti dall'NMT o dal protocollo dedicato al controllo degli errori. Il protocollo NMT si occupa di inviare messaggi che forzano la transizione di stato in questa macchina con messaggi formati da singoli frame di 2 byte;
- Protocolli speciali, comprendenti il SYNC protocol che controlla la sincronizzazione di tutti i dispositivi, il Time-Stamp protocol che assicura che i dispositivi lavorino allo stesso "Time of Day", infine l'Emergency protocol utilizzati dai dispositivi che presentano errori interni per informare gli altri elementi della rete;
- Protocolli per il controllo di errori tra cui Heartbeat protocol che verifica la disponibilità di tutti i partecipanti e il corretto stato della macchina a stati NMT.

Un esempio di rete dedicata che comunica utilizzando questo protocollo è quella che gestisce i public fan-coil nelle aree pubbliche. In questo caso, ogni area interessata è servita da un sistema che similmente alla rete CFCCS collega ogni utenza ad un dispositivo CANopen che prende il nome di Advantys STB, un assemblaggio di moduli I/O distribuiti, alimentazione e altro tipo che funzionano come nodo su una rete aperta del bus di campo. Consentono di progettare un'isola di 32 moduli I/O, distribuiti in modo da installare i moduli il più vicino possibile ai dispositivi meccanici da controllare. I moduli I/O possono essere interconnessi in gruppi chiamati *segmenti.* Ogni isola dispone di un segmento primario – il primo sul bus – il quale deve contenere almeno un modulo I/O di Advantys STB e contenere uno o più moduli di alimentazione in modo da distribuire l'alimentazione al resto dei moduli I/O. All'accensione dell'isola, ad ogni modulo che debba scambiare dati sul bus viene assegnato un indirizzo dal modulo di interfaccia di rete Advantys STB installato nel segmento primario, il quale potrà assegnare fino a 127 indirizzi in forma di numero intero, riservando per se stesso l'indirizzo 127.

Esiste un file ASCII, denominato EDS, che contiene informazioni sulle funzionalità di comunicazione del dispositivo e degli oggetti descritti nel suo dizionario oggetti, oltre agli oggetti specifici dei produttori. Tramite la configurazione di questo file si possono standardizzare gli strumenti per:

- Configurare dispositivi CANopen;

- Progettare reti per dispositivi CANopen;
- Gestire informazioni sui progetti da diverse piattaforme, tra cui Unity Pro.

L'isola Advantis STB richiede di esportare un EDS nel master del bus di campo, in questo caso specifico il PLC dell'unità di trattamento dell'aria a cui l'isola è associata. Così facendo l'EDS riassume il comportamento di tutta la rete come se fosse un singolo nodo, descrivendo al contempo tutti gli oggetti contenuti nel dizionario oggetti all'unità centrale di controllo.

4. Sviluppo del codice per l'automazione

4.1 Diagramma P&I

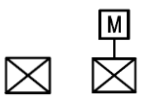
Visto il numero cospicuo di PLC impiegati nella realizzazione di questo impianto, per ognuno dei quali esiste un progetto dedicato, risulta impossibile trattare singolarmente ogni parte implementata su Unity Pro. Per cercare di rendere questo capitolo esaustivo si procederà quindi a studiare il processo decisionale e lo sviluppo in modo più generico, per studiare le fasi comuni a tutti i componenti del progetto che permettono comunque di dare una vista chiara e completa di tutti i dettagli fondamentali. Si descriverà per prima cosa lo studio della documentazione fornita dal cliente, che imporrà una serie di limiti di progettazione. Quindi si procederà con la descrizione della struttura e dei DFB Type comuni a tutti i dispositivi e si fornirà una breve descrizione di ogni sezione del task principale. Infine, verrà descritta la parte relativa ai dispositivi di sicurezza.

Il primo dei documenti forniti dal cliente è conosciuto comunemente come diagramma P&I, ovvero *Piping and Instrumentation Diagram*, che in italiano viene spesso denominato *schema di marcia*. È un disegno tecnico che riporta le interconnessioni tra tutte le apparecchiature di un processo e la strumentazione utilizzata per controllarlo. I simboli utilizzati per rappresentare le parti del sistema sono conformi agli standard ISA (Instrumentation, System and Automation Society), tuttavia si cerca di rendere la lettura del disegno più semplice e meno schematica utilizzando anche altri simboli. Nello specifico si riconoscono principalmente:

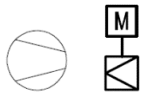
Sensori, trasmettitori, sonde e termostati;

Controllori;

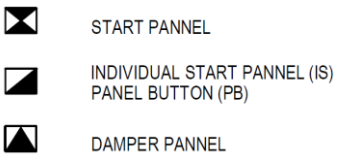
Pressure switch, ventole e frequency converter;



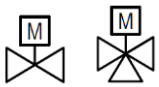
Damper a gravità o con attuatori, questi ultimi suddivisi in numerose categorie in base alla loro funzione e costruzione e rappresentati però dallo stesso simbolo per motivi pratici;



Compressore e regolatori di volume;



Pannelli di controllo;



Valvole a due o a tre vie;


Serpentine di riscaldamento o raffreddamento.

Essi rappresentano la maggioranza della strumentazione utilizzata in questo lavoro, insieme alle linee tratteggiate (vedi figura 4.1.1) che rappresentano i collegamenti elettrici.
Nei diagrammi, di cui la figura 4.1.1 ne rappresenta un estratto, vengono indicati i percorsi seguiti dall'aria, ed è interessante analizzare nello specifico le parti che li compongono:

- Ingresso, che comprende lo spazio dalla bocchetta di ingresso all'ingresso dell'unità di trattamento. Qui è presente un damper con attuatore.
- Trattamento in ingresso dell'aria, la parte di filtraggio ed eventuale condizionamento, ovvero la vera e propria unità di trattamento. Una parte di questa unità consiste nella cosiddetta *ruota entalpica,* un dispositivo scambiatore di energia che viene posizionato tra il flusso di aria in uscita e quello in entrata, i quali vengono fatti passare uno a fianco dell'altro nell'unità di trattamento proprio per permettere a questo scambiatore di recuperare parte dell'energia termica posseduta dall'aria in uscita per trasmetterla all'aria in ingresso. Dopo questo passaggio l'aria trattata è pronta per essere distribuita negli spazi a cui è destinata, spinta dalle ventole che provvedono a incanalarla nelle condotte.
- Condotte di mandata. Prima di arrivare nell'area da servire, l'aria passa attraverso una serie di sensori posti nelle tubature che hanno la funzione di dare un feedback al sistema riguardo la temperatura raggiunta dopo il trattamento.
- Area da servire. L'aria può essere utilizzata direttamente o può essere incanalata in dei fan-coil che permettono una ulteriore regolazione locale da parte dei passeggeri o di membri dell'equipaggio, per permettere ai passeggeri di avere in ogni momento le condizioni ambientali ideali.

- Condotte di ritorno, nelle quali sono spesso presenti damper necessari all'attuazione della smoke strategy oltre a sensori di vario tipo come sonde di rilevazione di CO2 o di misura di pressione;
- Trattamento in uscita, in base al tipo di area servita, l'aria in uscita viene estratta, filtrata ed eventualmente fatta confluire in una ruota entalpica;
- Uscita, ovvero il tratto che porta dall'unità di trattamento alla bocchetta che dà all'esterno, come in ingresso sono presenti dei damper con attuatori.

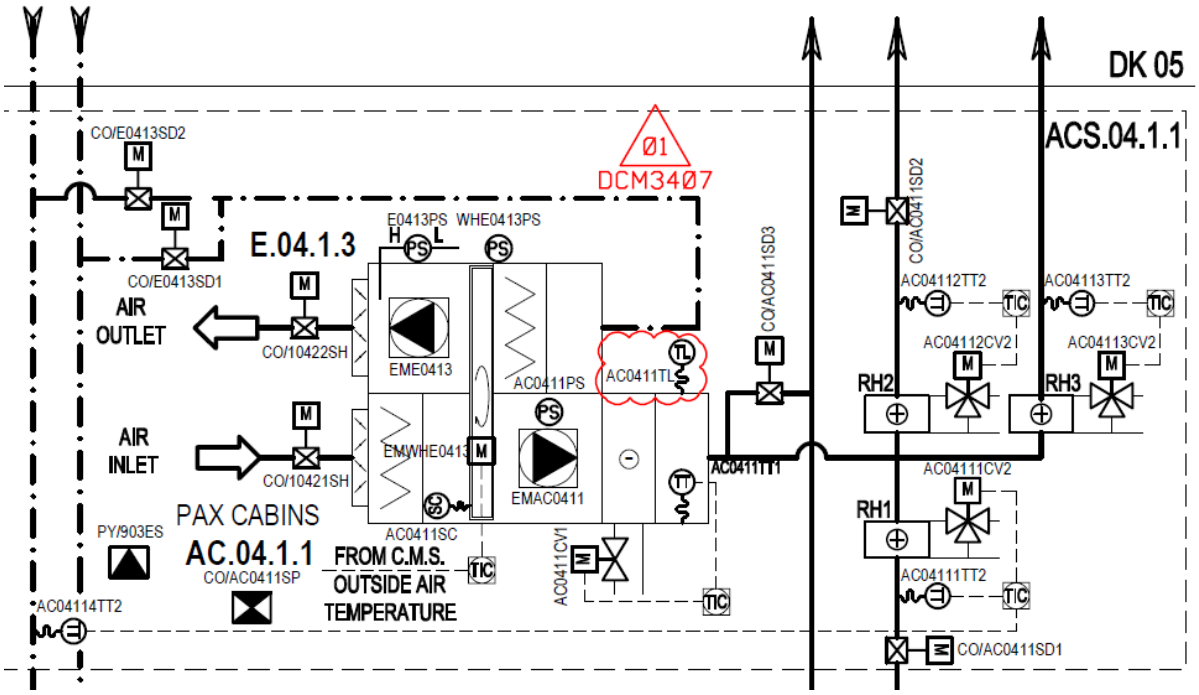


Figura 4.1.1 Estratto del P&I dell'area AC0411 che mostra il percorso dell'aria

4.2 DFB Type

Per attuare il controllo di ognuno degli elementi visti al paragrafo precedente e molti altri è possibile creare un DFB Type che restituisce in uscita dei comandi dopo avere elaborato gli stati dei dispositivi passatigli come parametri di ingresso. In questo paragrafo si cercherà di descrivere i DFB Type più importanti dividendoli in categorie.

La prima categoria di DFB Type descritta è quella che attua il controllo di quei dispositivi che forniscono in ingresso un solo dato. Questo può essere costante (regolato da SCADA) oppure filtrato e/o temporizzato (ad esempio la lettura di un sensore) . E ancora può essere memorizzato in un bit o in una word. Il punto comune di questi DFB Type è che presentano una sola sezione di codice divisa in generale in due parti: nella prima parte si elabora il dato grezzo e lo si passa alla variabile di uscita; nella seconda parte si gestiscono i comandi ricevuti da SCADA (tramite il segnale IO_PC).

- **S_FB_Ingresso**. Questo DFB Type serve a comunicare gli ingressi digitali non filtrati al software di supervisione. I parametri del blocco sono: in ingresso I_ST di tipo BOOL, corrisponde al segnale non elaborato inviato al PLC; in uscita O_PC_ST è il parametro in cui viene ricopiato il valore di ingresso che poi viene propagato al supervisore e O_PC_SIMU è il bit che indica lo stato di simulazione; nella sezione input/output IO_PC è la word di comando ricevuta dallo SCADA. È composto di una sola sezione chiamata **STATI**. In essa se non è attiva la modalità di simulazione si copia I_ST in O_PC_ST, a meno che non sia negato l'accesso; quindi, si registra lo stato dell'uscita in una variabile pubblica S_ST. Dopodiché si procede a gestire i comandi proveniente dal PC attraverso l'uso di un costrutto CASE. Tra i comandi possibili ci possono essere: accensione/spegnimento simulazione oppure accensione/spegnimento del segnale simulato. Infine, si resetta la word di comando IO_PC. All'interno del progetto questo DFB viene utilizzato per gestire gli ingressi provenienti da ESD e cappe da cucina.
- **S_FB_Ingresso_Ana.** Questo DFB Type ha lo scopo di analizzare le informazioni fornite dagli ingressi analogici e poi inoltrarle al supervisore. La struttura di ingressi e uscite è similare a quella vista per il DFB Type S_FB_Ingresso, con l'unica differenza che l'ingresso I_ST è di tipo INT anziché BOOL data la natura analogica del dato. Anche in questo caso è presente una sola sezione chiamata **STATI** suddivisa in tre parti: elaborazione delle uscite, gestione dei comandi inviati dal sistema SCADA, eventuale scala di alcuni dati. Il processo di acquisizione del dato consiste nell'accumulo in una variabile interna del dato fornito dalla word di ingresso I_ST per un certo numero di cicli macchina; quindi, viene calcolata la media dividendo il risultato nell'accumulatore per il numero di cicli eseguiti e si ottiene così il dato da passare al supervisore. In caso lo SCADA lo richiedesse, il dato può essere scalato per avere la proporzione più adatta. Questo DFB Type viene associato agli ingressi provenienti da trasmettitori di temperatura, di pressione e rilevatori di CO2.
- **S_FB_Sensore.** Questo DFB Type serve a gestire i dati provenienti da sensori che inviano dati di tipo BOOL. La struttura è simile a quella di S_FB_Ingresso; tuttavia, questo tipo deve gestire informazioni provenienti da un flusso continuo e con carattere impulsivo, può azionare degli allarmi e temporizzare l'acquisizione. I parametri di ingresso e uscita prevedono, oltre a quelli visti per S_FB_Ingresso, i parametri di allarme O_PC_PRE che identifica un preallarme non ritentivo e O_PC_ALL che consiste nel vero e proprio allarme ritentivo. L'unica sezione è denominata **STATI** ed è divisa in: elaborazione degli stati di uscita, temporizzazione del segnale e del

preallarme, reset degli allarmi e gestione dei comandi ricevuti. Il processo prevede che l'uscita venga aggiornata ogni cento millisecondi. Se il sensore mantiene attiva la propria uscita, il codice mantiene al valore TRUE la variabile di uscita O_PC_ST, quando I_ST torna al valore 0V allora anche l' uscita viene modificata e viene portata a FALSE. All'interno del progetto questo blocco viene utilizzato per regolare i *pressure switch* e i *termostati di temperatura massima.*

- **S_FB_Soglia.** Questo DFB Type è stato creato con lo scopo di gestire il superamento da parte di un ingresso di una certa soglia regolata da SCADA. La struttura è simile a quella di S_FB_Sensore, infatti, contiene tutti i parametri introdotti in quel blocco e in più contiene il parametro di ingresso I_SOGLIA di tipo INT che specifica un valore limite massimo o minimo che il valore misurato I_ST – anch'esso di tipo INT – non può oltrepassare. L'unica sezione presente è denominata STATI che si suddivide in: elaborazione delle uscite, temporizzazione e conteggio dell'allarme, reset dell'allarme e gestione dei comandi provenienti dal supervisore. L'uscita O_PC_ST è di tipo BOOL in quanto si vuole notificare al supervisore solo se il valore misurato è inferiore o superiore alla soglia. Per effettuare la comparazione del valore misurato con la soglia viene considerato anche un certo grado di isteresi, in modo da includere un certo grado di incertezza ma anche per assicurare la stabilità in caso il dato passi da un processo di regolazione, in quanto regolare una grandezza ad un livello esatto risulta di fatto impossibile. In caso di superamento di questa soglia e di una certa indicazione dal supervisore si fa partire il contatore collegato alla *timebase* %S5 – corrispondente ai 100 millisecondi - che in base al valore raggiunto azionerà a sua volta l'uscita corrispondente al preallarme e all'allarme, a meno che non venga azzerato prima. Il reset degli allarmi in questo caso è consentito solo attraverso il comando del supervisore. Infine, si gestiscono i comandi per la simulazione. In questo progetto questo DFB Type viene utilizzato per le soglie minime e massime dei misuratori di temperatura posti sia in ingresso che in uscita alle aree servite dai fan-coil, per le soglie massime e minime dei trasmettitori di umidità posti nelle aree servite dai fan-coil, per le soglie massime e minime dei trasmettitori di pressione posti in prossimità delle ventole, per le soglie massime e minime di tutti i trasmettitori di temperatura posti a monte delle aree servite dalla AHU e infine per la soglia massima dei rilevatori di anidride carbonica.
- **S_FB_Comando.** In questo DFB Type sono contenute le istruzioni per attuare l'attivazione logica di un'uscita di tipo BOOL. I parametri di ingresso sono compresi

tra le variabili pubbliche. Nello specifico si sottolinea la variabile K.CMD, contenuta nella struttura K (DDT), mentre i parametri di uscita sono il bit di uscita O_RQ, il bit di uscita per la variabile SCADA e il bit di gestione dell'uscita in manuale (ovvero dal PLC). Come input/output è presente solo il comando IO_PC. L'unica sezione di questo DFB Type è denominata **STATI** e prevede una parte di assegnazione delle variabili di uscita e una parte di gestione dei comandi proveniente da SCADA. L'uscita O_RQ viene posta al valore 1 se il comando proveniente da PLC tramite struttura K lo prevede, oppure tramite forzatura in fase di comando manuale. In questo impianto questo DFB Type verrà utilizzato per istanziare il comando di attivazione della *smoke strategy* (inviato dal PLC quindi attivato se l' emergenza proviene dall' ESD della stessa stazione) per le unità di trattamento che lo prevederanno.

I seguenti DFB Type a differenza dei precedenti descrivono il controllo di dispositivi con un comportamento generalmente più complesso. Sono caratterizzati da un numero maggiore di ingressi e uscite; perciò, le sezioni di codice sono divise in più parti (quasi sempre divise tra PC, STATI, USCITE e ALLARMI) in modo da semplificare la lettura del programma e rendere lo sviluppo sistematico.

- **Comandi_Stazione**. È il DFB Type necessario per far sì che si possa comandare una stazione di controllo di una macchina dal PC di supervisione. I parametri di ingresso e uscita sono commentati nell'immagine 4.2.1. Sostanzialmente gli ingressi riepilogano le modalità di funzionamento della macchina, gli ingressi di ESD e allarmi e lo stato dei motori; le uscite consistono in cinque word di stato da inviare al supervisore.

Comandi_Stazione		<DFB>		Rev. 1 del 05-02-2010 ETA
<inputs>				
I_LOC_REM	1	BOOL		Local/Remote Mode (0 = AHU in Local; 1 = AHU in Remote)
I_SUMM_WINT	2	BOOL		Summer/Winter Mode (0 = Summer; 1 = Winter)
I_COMM	3	BOOL		Ethernet Communication Status (0 = Communication OK; 1 = C...
I_SC_MODE	4	BOOL		SC MODE (ingresso per stazioni con Inverter)
I_ESD	5	ANY_ARRAY_EBOOL		Ingressi relativi ad ESD di Stazione
I_SYS	6	ANY_ARRAY_EBOOL		Ingressi relativi ad allarmi di Stazione
I_AUX	7	ANY_ARRAY_EBOOL		Ingressi AUX che prevedono solo Segnalazione (Halton)
I_ESD_NO_ALM	8	ANY_ARRAY_EBOOL		Ingressi ESD cablati (es. Start Stairs Fan)
I_ST_MOT	9	BOOL		Cumulativo Stato Running Motori Stazione
I_ST_EW	10	BOOL		Stato Running Ruota Entalpica
I_ST_MANU	11	BOOL		Cumulativo Stato Manuale Motori/Serrande Stazione
<outputs>				
O_PC_ST	1	INT		Word di Stato
O_PC_ST_ESD	2	INT		Word di Stato per ESD
O_PC_ST_AUX	3	INT		Word di Stato per AUX
O_PC_ST_ESD_NO_ALM	4	INT		Word di Stato per ESD cablati
O_CAN_STATUS	6	INT		

Figura 4.2.1 Parametri di ingresso e uscita del DFB Type Comandi_Stazione

Sono previste quattro sezioni di codice: **PC** pone la macchina nello stato indicato da un comando inviato da remoto (IO_PC). **USCITE** è un breve riassunto delle uscite scritte in linguaggio ladder. **STATI** è la sezione in cui l'AHU comunica il proprio stato allo SCADA. Prima compilando la word O_PC_ST, quindi associando gli ingressi fisici (%I) dei segnali ESD a un indirizzo della memoria di sistema (%MW) usando la funzione MOVE_ARBOOL_INT e, infine, notificando lo stato della comunicazione tra la stazione e i device Advantys connessi tramite rete CANopen. **ALLARMI** verifica la presenza di segnali attivi proveniente da ESD interni alla stazione, ingressi ESD cablati nella stazione (ad esempio quelli delle aree nelle sue vicinanze) o allarmi provenienti dalla stazione stessa. La verifica avviene ispezionando gli array predisposti a raggruppare questi segnali (I_ESD, I_SYS e I_ESD_NO_ALM). In questo progetto questo DFB viene usato per comandare le stazioni delle unità di trattamento dell'aria.

- **S_FB_Inverter.** Questo DFB Type serve a gestire il funzionamento degli inverter. I parametri di ingresso descrivono il comportamento corrente del dispositivo, includendo la velocità corrente, modalità di funzionamento locale o remota, By-pass e bit cumulativo degli allarmi. I parametri di uscita comprendono i comandi da inviare al dispositivo (marcia diretta, marcia inversa, arresto preventivo, setpoint di velocità) e gli stati da inviare al supervisore (stato, stato ausiliario, setpoint e contatori). È utile notare che dall'interfaccia HMI si possono inviare comandi tramite il segnale IO_PC oppure impostare manualmente un setpoint di velocità tramite il segnale IO_PC_SET. Le sezioni del blocco sono quattro: **PC** in cui si verifica l'attivazione del parametro IO_PC e tramite un costrutto CASE si verifica il tipo di comando che viene ricevuto. I comandi si dividono in comandi per il funzionamento e comandi di simulazione e ad ognuno corrisponde la regolazione di un parametro di uscita o una variabile interna. Alla fine, IO_PC viene azzerato. La sezione **USCITE** permette di attivare il comando di marcia diretta o inversa, sotto la condizione di avere attivato il comando manuale o il bypass della macchina; inoltre, questa sezione permette di attivare il flag comando in corso W_EXEC, che viene utilizzato per controllare alcuni allarmi. La sezione **ALLARMI** si occupa di resettare il flag W_EXEC quando la parola di stato è stata aggiornata correttamente, in caso contrario attiva gli allarmi di mancato avvio/fermata; se anche uno di questi risulta positivo viene attivato il flag di allarme cumulativo nella parola di stato. Infine, viene gestito il reset che viene dato dal comando apposito del PLC o manualmente dal pannello operatore. La sezione **STATI** gestisce

l'interfacciamento verso un inverter cablato direttamente al PLC o con collegamenti seriali. Attraverso la struttura di stato interno denominata S si regolano i valori delle variabili che vengono successivamente specchiate nella parola di stato da comunicare all'interfaccia O_PC_ST, inoltre vengono anche specificati i valori dei bit 0-4 della parola di stato ausiliaria O_PC_AUX utilizzati nella sezione allarmi. Il codice prosegue con il conteggio dei minuti/ore di marcia, reset comandi di allarme o emergenza, incremento di alcuni contatori collegati allo stato di allarme, e infine viene regolato il set point di velocità. Soffermandoci su quest'ultimo notiamo che può essere settato in modalità manuale da HMI oppure in automatico assegnandogli il valore K_VEL prestabilito nella sezione apposita (MAST -> GESTIONE_INVERTER). Nel momento in cui si passa da automatico a manuale i comandi vengono inizialmente copiati con lo stesso valore. Questo DFB Type viene utilizzato nel progetto per gestire le istanze di inverter che vengono collocati nelle AHU oppure nelle reti CANopen a gestire i *public fancoil*.

- **S_FB_Motore.** Questo DFB Type serve a gestire i motori elettrici monofase. I parametri di ingresso sintetizzano le informazioni date dal motore al PLC riguardanti il funzionamento diretto o a due velocità, lo stato del *sezionatore* e della protezione termica, il cumulativo degli allarmi, il flag di bypass e il selettore per il comando locale. I parametri di uscita sono i comandi di marcia (diretto e a due velocità), le parole di stato e di stato ausiliario e lo stato dei contattori espresso in un array. Le sezioni sono le stesse del DFB Type Inverter e svolgono un compito simile, se non contiamo il controllo della velocità. All'interno del progetto, questo DFB Type viene utilizzato per istanziare tutti i motori elettrici che non necessitano di regolatori di velocità situati all'interno di unità di trattamento dell'aria o in prossimità della bocchetta di uscita/entrata dell'aria.
- **S_FB_PID.** Questo DFB Type ha il compito di gestire il funzionamento dei controllori PID. I parametri di ingresso comprendono l'ingresso di regolazione (I_REG), i coefficienti proporzionale P integrativo I e derivativo D, i valori soglia minimi e massimi impostati da SCADA. I parametri di uscita sono l'output di regolazione (O_REG), le parole di stato e di stato ausiliario e alcuni valori di appoggio per lo SCADA in cui copiare i valori di regolazione in ingresso, in uscita e il setpoint fissato dal PLC. Le sezioni di codice sono quattro. **PC** gestisce i comandi provenienti da HMI. **USCITE** si occupa di assegnare un valore a tutti i parametri fissi (figura 4.2.2) e impostare le soglie di regolazione. In questa sezione avviene anche il processo di regolazione vero e proprio implementato dalla funzione integrata PID_FF (figura

4.2.3) e i successivi controlli che il parametro di uscita OUTD non superi le soglie. Una volta calcolati tutti i parametri, essi vengono assegnati alle uscite corrispondenti. **ALLARMI** si occupa di attivare gli allarmi quando le variabili di regolazione superano certe soglie, oppure di resettarli quando il PLC propaga il relativo comando. Infine, **STATI** si occupa di comunicare lo stato del dispositivo controllato al supervisore tramite la regolazione del parametro O_PC_ST (figura 4.2.4).

```
(*Impostazioni parametri fissi*)
(*W_PID_PARAM[8].8:=1;*)
W_PID_PARAM.bump:=0;
(*W_PID_PARAM[5]:=10;*)
W_PID_PARAM.pv_inf:=0.0;
W_PID_PARAM.pv_sup:=10000.0;
W_PID_PARAM.out_inf:=W_CFG_OUT_MIN;
W_PID_PARAM.out_sup:=10000.0;
W_PID_PARAM.out_min:=INT_TO_REAL(W_CFG_SL_MIN);
W_PID_PARAM.out_max:=INT_TO_REAL(W_CFG_SL_MAX);
W_PID_PARAM.dband:=CFG_DBAND;
W_PID_PARAM.gain_kp:=CFG_GAIN_KP;
W_PID_PARAM.outrate:=CFG_OUTRATE;
W_PID_PARAM.outbias:=0.0;
W_PID_PARAM.rev_dir:=CFG.2;                    (*0:azione diretta; 1:azione inversa*)
```

Figura 4.2.2 Assegnazione dei parametri fissi del regolatore PID

PID_FF		PIDFF		
<inputs>				
PV	1	REAL		Process value
SP	2	REAL		Setpoint
FF	3	REAL		Deviation input
RCPY	4	REAL		Actuator position copy
MAN_AUTO	5	BOOL		Manual (0) or automatic (1) mode
PARA	6	Para_PIDFF		Parameter
TR_I	7	REAL		Initialization input
TR_S	8	BOOL		Initialization command
<outputs>				
OUTD	1	REAL		Output variation from PID
MA_O	4	BOOL		Automatic mode (1) or other mode (0)
INFO	5	Info_PIDFF		Information
STATUS	6	WORD		Status word
<inputs/outputs>				
OUT	9	REAL		Absolute value output

Figura 4.2.3 Parametri di ingresso e uscita della funzione integrata PID_FF

```
(* ------------- *)
(* STATI A SCADA *)
(* ------------- *)
O_PC_ST.0:=S.OK AND NOT I_LOC;          (*In regolazione*)
O_PC_ST.1:=W_SET_MINIMO;                (*Regolazione Limitata al minimo*)
O_PC_ST.2:=W_SET_MASSIMO;               (*Regolazione Limitata al massimo*)
O_PC_ST.8:=I_LOC;                       (* ATTUATORE COMANDATO LOCALMENTE *)
O_PC_ST.9:=CFG.2;                       (* MODALITA' DI REGOLAZIONE *)
O_PC_ST.10:=W_K_PC_MAN_SET;             (* Stato di SET POINT in manuale*)
O_PC_ST.11:=W_K_PC_AUT_PID;             (* Stato di USCITA in AUTOMATICO*)
O_PC_ST.12:=W_K_PC_MAN_PID;             (* Stato di USCITA in MANUALE*)
O_PC_ST.13:=S.ALL;                      (* STATO DI CUMULATIVO ALLARMI PER SCADA *)
O_PC_ST.14:=S.IN_USO:=K.IN_USO;         (* UTENZA IN USO *)
```

Figura 4.2.4 WORD di stato da inviare allo SCADA

- **S_FB_Deviatrice.** Questo DFB Type ha il compito di gestire il comportamento di deviatrici, serrande ed elettrovalvole. I parametri di ingresso comprendono gli stati di fine corsa aperta(I_POS_1) e chiusa(I_POS_0) e il bit di comando locale/remoto. I

parametri di uscita comprendono i segnali di uscita *aperto* e *chiuso* (di tipo BOOL), la parola di stato da trasmettere al supervisore e dei contatori disposti in un array. Le sezioni sono quattro. **PC** serve per la gestione dei comandi ricevuti da HMI tra cui: Comando posizione chiusa/aperta, reset del bit *utenza in uso*, comando di funzionamento automatico/manuale, comandi di simulazione ON/OFF, reset allarmi e azzeramento conteggi. **USCITE** descrive in linguaggio Ladder le sequenze necessarie al passaggio di stato da fine corsa chiuso a fine corsa aperto. **ALLARMI** gestisce gli allarmi del dispositivo, che prendono il nome di allarmi di mancato posizionamento. Questi allarmi vengono attivati quando il contatore apposito supera la soglia definita dal parametro CGF_T_COMM che definisce il tempo di commutazione del dispositivo. Un altro allarme è dato dal superamento dello stesso counter della soglia definita dal parametro CFG_T_INT, indice della presenza della stessa incongruenza di posizionamento. In presenza di uno di questi tre allarmi, viene settato il flag di allarme cumulativo. Il reset degli allarmi può provenire solo dall'apposito comando del PLC. Infine, il software tiene conto del numero di posizionamenti andati a buon fine o terminati con l'attivazione di un allarme, attraverso l'uso di due variabili di accumulazione dedicate. **STATI** provvede alla corretta formazione della parola di output da inviare al supervisore O_PC_ST. In questo progetto il DFB Type viene impiegato per attuare il controllo di *damper motorizzati*.

- **S_FB_VAV.** Questo DFB Type serve a gestire i cosiddetti variatori di volume d'aria. I parametri di input comprendono il dato rilevato di CO2, le soglie massime e minime di CO2 ppm (per soglia minima di CO2 si intende il valore al di sotto del quale i rischi per esposizione sono nulli), i valori percentuali di apertura della valvola, i quali vengono copiati in uscita in caso di superamento di tali soglie (si nota che questi valori vengono chiamati O_PERC_MIN/MAX anche se sono parametri di input), il comando di controllo manuale, il comando di abilitazione e infine I_REF_FORCED che serve a gestire lo stato del dispositivo in funzione dello stato di altre valvole VAV. L'unico parametro di uscita è O_RQ che consiste nella parola da inviare al dispositivo la quale specifica la percentuale di apertura della valvola. Le sezioni di codice sono tre. **USCITA** si occupa di assegnare il corretto valore al parametro di uscita in base al fatto che esso venga comandato in automatico o in manuale. **STATI** si occupa di elaborare il dato rilevato di CO2 in modo da effettuare un controllo lineare dell'apertura delle valvole. Si distinguono due scenari principali: se la valvola non è abilitata(I_EN=0), essa considererà solo il segnale di forzatura (I_REF_FORCED) che verrà direttamente passato al parametro di uscita; se la valvola è abilitata (I_EN = 1)

allora procederà all'elaborazione del dato rilevato di CO2. Il controllo lineare verifica se il dato rilevato è minore della soglia minima e in quel caso assegna alla variabile di appoggio il valore stabilito O_PERC_MIN, poi verifica se ha superato la soglia massima e in quel caso gli assegna O_PERC_MAX. In caso il valore sia compreso tra le due soglie avviene il vero e proprio controllo lineare: considerando la funzione che vede come varia la percentuale di apertura rispetto al dato rilevato di CO2, possiamo calcolare il coefficiente angolare della retta passante per O_PERC_MIN e O_PERC_MAX; così facendo ricaviamo l'equazione della retta e sostituendo il dato rilevato di CO2 al suo interno si ricava facilmente il valore corretto da assegnare all'output. Segue una verifica che il valore dell'uscita non assuma valori negativi o superiori al 100%. Nel progetto in questione questo tipo di DFB viene usato per creare istanze di particolari serrande dette *regolatori a volume variabile.*

I seguenti quattro DFB Type sono stati creati per realizzare il controllo di dispositivi con caratteristiche meno generali di quelli descritti finora.

- **ATV_MNG**. Questo DFB Type serve a comandare gli inverter Altivar collegati alla AHU. I parametri di ingresso sono: il bit che segnala il funzionamento in modalità standard SC_MODE, la word di stato del motore SW, il bit di accensione RUN_STOP, il timebase di sistema CLOCK (appoggiato a %S6), il bit che indica lo stato del bus di comunicazione e il ritorno fisico di avvio delle ventole. I parametri di uscita sono i tre bit che individuano gli stati READY, RUN, FAULT del motore e la command word CW. Il codice è scritto in linguaggio ladder ed è diviso in due sezioni: **COMANDI** si occupa di gestire lo stato dell'inverter in base ai comandi proveniente dal PLC e dallo SCADA; **SIMULAZ** serve ad assegnare lo stato iniziale all'inverter in caso di attivazione della simulazione. Gli output sono gli stati dell'inverter e la *command word* CW, ovvero la parola contente tutti i comandi destinati ai fan-coil gestiti dallo stesso inverter.

- **FB_Ruota.** Questo DFB Type è stato creato per gestire il funzionamento di ruote entalpiche. I parametri di ingresso contengono informazioni riguardanti il funzionamento corrente dell'apparecchio, le condizioni climatiche esterne, i *flag* di allarme, l'abilitazione della modalità inverno/estate e altre. I parametri di uscita comprendono il risultato del calcolo delle rotazioni per minuto della macchina, il flag

che segnala il guasto del rilevatore di velocità e il comando di start/stop. È utile descrivere anche la struttura privata CTU_EW_SD del tipo elementare *contatore di interi* (CNT_INT) che attraverso l'impostazione di un trigger permette il conteggio di una nuova unità da sommare al valore di conteggio CV, inoltre permette di impostare una variabile di reset R e un valore iniziale di conteggio PV. Le sezioni sono quattro, scritte in linguaggio Ladder come richiesto dal cliente. **USCITE** gestisce il *set* e il *reset* del comando di avvio della macchina. **RPM** calcola le rotazioni per minuto grazie ad un sensore. Nel momento in cui questa rilevazione dovesse venir meno, viene riavviato il contatore di rotazioni. Viene controllato inoltre che il valore delle rotazioni non superi il valore 90 per minuto e in caso si aziona l'allarme O_ALM. Tale allarme porta all'azzeramento del conteggio in alternativa allo spegnimento della macchina. **OP_CONDITION** controlla i flag di abilitazione del funzionamento della macchina. Tali abilitazioni dipendono sostanzialmente dal superamento delle temperature di soglia estiva o invernale impostate dal personale della nave. Infine, **AUTO_CHECK** gestisce il funzionamento automatico della ruota. Nello specifico si fa sì che in modalità automatica la ruota opera per due minuti ogni ora di funzionamento dell'unità di trattamento dell'aria a meno che non venga specificato diversamente.

- **FB_Elec_Heater.** Questo DFB Type serve a gestire il funzionamento di scaldabagno elettrici. I parametri di ingresso comprendono il comando di abilitazione, il range di temperature raggiungibili, il valore di *set point*, il valore di *isteresi*, il numero di livelli realizzabili in uscita – un valore costante posto a 4 - e il valore registrato in ingresso. In uscita troviamo il valore di uscita di regolazione e il valore massimo di uscita. Per implementare il funzionamento del dispositivo viene usata una sola sezione, **USCITE**. Per prima cosa si calcola l'ampiezza dei livelli di temperatura, dividendo il RANGE per il numero di livelli raggiungibili in uscita. Quindi con un ciclo *while* rappresentato in figura 4.2.5, si misura di quanto l'ingresso si discosta dal set point, considerando anche il valore di isteresi e in base a questo valore viene regolata l'uscita. Si nota che il valore di uscita cresce esponenzialmente rispetto alla distanza tra ingresso e set point, fino al valore limite OUT_MAX. Nel momento in cui il setpoint viene raggiunto il dispositivo viene spento. Con questo DFB Type si vuole comandare gli scaldabagno collocati per la maggior parte all'interno delle cabine dei passeggeri e dell'equipaggio.

```
IF ENABLE and I_REG <= (SP+ISTERESI) THEN

        pow_i:=0;
        i:=0;

        (* iterazione sul numero di livelli d'uscita *)
        WHILE i<N_LIV_OUT DC

                pow_2:=1;
                j:=0;
                (* calcolo la potenza di 2 *)
                WHILE j<i DC
                        pow_2:=pow_2*2;
                        j:=j+1;
                END_WHILE;

                pow_i:=pow_i+pow_2;
                (* verifica gradino *)
                if I_REG <= (SP - i*n_liv_x - ISTERESI) and (EL_HEATER < pow_i) then
                        (* assegnazione livello *)
                        EL_HEATER:=pow_i;

                elsif I_REG >= (SP - (i+1)*n_liv_x + ISTERESI) and (EL_HEATER >= pow_i) then
                        (* assegnazione livello *)
                        EL_HEATER:=pow_i;

                end_if;
                i:=i+1;
        END_WHILE;


        OUT_MAX:=pow_i;

        ELSE

        EL_HEATER:=0;

END_IF;
```

Figura 4.2.5 processo di regolazione della potenza di erogazione dei fancoil

- **FB_Timebands.** Questo DFB Type serve per gestire le AHU che hanno bisogno di funzionare solo in certe fasce orarie. I parametri di ingresso sono il tempo attuale fornito da SCADA e gli orari di inizio e fine di tre diverse fasce orarie. I parametri di uscita sono O_TIME_REQ, per aggiornare l'orologio ogni minuto e O_NEW_SPEED, un bit che viene attivato ogni qualvolta viene cambiata la velocità di funzionamento. Le sezioni di codice ne comprendono una per ogni fascia oraria e una per il comando delle uscite chiamata **VAR_SPEED_ACT**. Il funzionamento di ogni fascia oraria è equivalente e consiste nel semplice controllo che il tempo corrente sia compreso tra l'intervallo che definisce la fascia oraria. In caso positivo viene settato il bit di appoggio che nella sezione apposita verrà copiato nel bit di uscita O_NEW SPEED. Tutto ciò a valle dei controlli che per tale unità siano stati definiti gli intervalli orari.

Ulteriori DFB Type sono necessari per la corretta implementazione della comunicazione con le reti accessorie viste nel capitolo 3. Prima di passare alla loro descrizione è utile vedere alcune strutture dati usate in molti di essi.

CF_DATA è la struttura rappresentata in figura 4.2.6, che raggruppa le informazioni riguardanti il corrente funzionamento del cabin fancoil e tutta una serie di informazioni sulla cabina che questo va a servire.

Name	Type	Comment
CF_DATA	<Struct>	
SPD	INT	Fan speed
RST	INT	Room supply temperature
ART	INT	Actual room temperature
EHC	INT	Electric heater command
CVC	INT	Cooling valve command
RSP	INT	Room temperature setpoint
RSPMIN	INT	Difference between central setpoint line and min. Setpoint
RSPMAX	INT	Difference between central setpoint line and max. Setpoint
EHS	BOOL	Electric heater safety switch
BDS	BOOL	Balcony door switch
KCS	BOOL	Key card status
AFS	BOOL	Air filter status
COM	BOOL	Errore comunicazione
FSMANU	BOOL	Fan Speed manual / auto 1= manual
EHMANU	BOOL	El. Heater manual / auto 1= manual
CVMANU	BOOL	Cooling valve manual / auto 1= manual
ONOFF	BOOL	Fan coil On/Off command 1 = ON
PSEL	BOOL	Cabin sold for 2/4 passenger selection 0=2 passengers, 1=4 passengers
ESE	BOOL	Energy Strategy enable/disable 1= enabled
ESCMDB	INT	Energy saving cooling mode dead band (dead band above actual setpoint)
ESHMDB	INT	Energy saving heating mode dead band (dead band below actual setpoint)

Figura 4.2.6 CF_DATA Derived Data Type

DT_GESTIONE identifica un DDT (*Derived Data Type*) che definisce una struttura dati che contiene due variabili: JOB_ACT è il numero di job da eseguire, EN_JOB è il bit che abilita alla richiesta del job. Si può pensare ad un job come l'identificativo del tipo di operazione che si vuole effettuare. Essendo questo DDT utilizzato soprattutto in funzioni per la comunicazione si evince che le operazioni in oggetto ricadono per lo più in questo ambito.

TAB_COM è un array messo a disposizione dall'ambiente di sviluppo. Contiene le quattro parole per la *gestione della comunicazione* ovvero quei parametri usati per gestire le funzioni di comunicazione di ogni tipo all'interno dell'ambiente di sviluppo. Le prime due parole sono gestite dal sistema mentre le ultime due dall'utente. La struttura dell'array è mostrata in tabella:

NUMERO PAROLA	BYTE PIÙ SIGNIFICATIVO	BYTE MENO SIGNIFICATIVO
TAB_GEST[0]	Numero di scambio	BIT 0: bit di attività BIT 1: bit di cancellazione BIT 2: bit di acknowledgment
TAB_GEST[1]	Report di operazione	Report di comunicazione
TAB_GEST[2]	time-out	
TAB_GEST[3]	length	

Dove:

- **Bit di attività:** è il bit che indica lo stato di esecuzione della funzione di comunicazione. È posto a 1 quando viene inviata e 0 quando viene completata;
- **Bit di cancellazione:** è il bit usato per interrompere funzioni di comunicazione in corso;
- **Bit di Acknowledgment;**
- **Numero di scambio:** è il numero assegnato dal sistema che individua univocamente la funzione di comunicazione inviata. Viene usato ove necessario per fermare lo scambio in corso (usando il bit di cancellazione);
- **Report di comunicazione:** è un parametro che assume significato quando avviene la transizione da 1 a 0 del bit di attività. Il valore a esso assegnato indica in che modo è terminata la comunicazione (ad es. 16#00 indica scambio avvenuto correttamente);
- **Report di operazione:** è un parametro che assume significato quando il report di comunicazione assume il valore 16#00 (comunicazione avvenuta correttamente) o 16#FF (messaggio scartato). La sua funzione è quella di specificare i risultati dell'operazione avvenuta sull'applicazione remota;
- **Time-out:** tempo massimo di attesa della risposta. Questo parametro è appoggiato alla timebase 100ms (%S5); il sistema scarta ogni risposta arrivata dopo il tempo massimo;
- **Length:** indica il numero di byte del messaggio.

- **CF_Read e CF_Public_Read.** Questo DFB Type gestisce la lettura dei dati forniti dai *cabin fancoil*. I parametri di ingresso comprendono: la struttura I_MB_GEST di tipo DT_GESTIONE; l'array di stringhe I_ADDR in cui ogni stringa è la concatenazione dell'indirizzo IP del dispositivo EGX e l'indirizzo CANopen, creando così l'indirizzo Modbus; infine I_WRITE è il bit che segnala che il master sta scrivendo sul dispositivo o in generale sta utilizzando il bus. L'unico output è l'array DATA che raggruppa le informazioni raccolte da tutti i fancoil presenti sulla rete. Si nota che sulla rete sono supportabili fino a 30 fancoil. Si nota che CF_PUBLIC_READ viene istanziato solo sul primo dei fancoil mentre in tutti gli altri si usa CF_READ. Il codice relativo al DFB Type è diviso in tre parti. Nella prima parte si calcola l'indice lettura in funzione del numero di CF da leggere, nella seconda parte si leggono i registri corrispondenti al tipo di job da eseguire, infine nella terza parte si controlla se la lettura è andata a buon fine o meno: nel primo caso si copiano i dati nell'array di output DATA, altrimenti si setta un bit di errore nell'array di interi COM_ERROR

all'elemento corrispondente al CF_TO READ, bit 2 e si incrementa il contatore di errori COUNTER_ERROR corrispondente al CF_TO_READ.

- **GEST_COMM.** Questo DFB Type si occupa di regolare i valori delle strutture DT_GESTIONE e TAB_COM, le quali si influenzano a vicenda. I parametri di ingresso comprendono il time-out di comunicazione, N_FC specifica il numero di utenze da leggere – ovvero il numero di operazioni lettura/scrittura - T_DELAY è il tempo che deve trascorrere tra la conclusione di un job e la richiesta del successivo, I_WRITE è l'ingresso che abilita la scrittura. I parametri input/output sono l'array TAB_COM contenente le quattro word per la gestione comunicazione e la struttura GEST, di tipo DT_GESTIONE. È presente una sola sezione di codice scritta in linguaggio Ladder **GESTIONE**. Dal bit TAB_COM[0].0 a zero si setta EN_JOB tramite un TON. L'abilitazione di questo bit permette di operare sugli elementi dello stesso array TAB_COM e incrementare il JOB_ACT. In seguito, controllando il segno della variabile di ingresso FOCUS_ON_FC si aggiornano i valori JOB_MIN e JOB_MAX che corrispondono ai limiti inferiori e superiori del range di job da eseguire. Tali valori sono definiti dal programmatore in base al contesto.
- **READ_DATA.** Questo DFB Type è usato per leggere i registri dei dispositivi EGX e popolare le tabelle di ogni cabin fancoil. I parametri di input e output sono commentati nell'immagine 4.2.7. Si sottolinea MB_CODE che indica il tipo di dato scambiato. La sezione di codice READ mostra come in realtà la logica dietro a questa funzione si avvicina molto a quella della funzione elementare READ_VAR. La funzione in questione prende in ingresso l'indirizzo di un dispositivo e le locazioni dei suoi registri da leggere per depositare una copia del loro contenuto nell'array apposito di uscita. In aggiunta a questa procedura, il blocco si occupa di operare su variabili pubbliche per notificare al sistema una serie di statistiche sulle letture e comanda le uscite aggiuntive.

READ_DATA		<DFB>		Blocco funzione "specializzato" nella lettura di word
<inputs>				
ADR_DEV	2	ADDM_TYPE		Indirizzo del dispositivo da leggere
REG_SOURCE	3	DINT		Registro iniziale dei dati da leggere
N_REG	4	INT		Numero di registri da leggere
JOB_ITEM	5	INT		N° del Job da eseguire
MB_CODE	7	STRING		
<outputs>				
JOB_OK	2	BOOL		Operazione conclusa positivamente (BLINK)
JOB_ERROR	3	BOOL		Operazione conclusa con errore
DATA_RECV	5	ANY_ARRAY_INT		Dati ricevuti

Figura 4.2.7 Parametri di ingresso e uscita del DFB Type READ_DATA

- **READ_CF.** Questo DFB Type ha lo scopo di andare a leggere i dati scritti nei registri dei dispositivi EGX in modo da popolare la struttura CF_DATA utilizzando il DFB

Type READ_DATA. I parametri di ingresso includono IP_MB_ADDR, l'indirizzo IP del dispositivo Ethernet EGX e JOB_ID_INITIAL che indica la tipologia di compito da eseguire. In uscita viene restituito un'istanza della struttura CF_DATA. La sezione di codice, chiamata anch'essa READ_CF è scritta in linguaggio FBD. In essa l'indirizzo IP viene convertito tramite la funzione ADDM da stringa a indirizzo utilizzabile direttamente in uno dei quattro blocchi READ_DATA successivi. Questi blocchi hanno la funzione di leggere dei dati contenuti nei registri dei dispositivi EGX e di trasferirli in apposite tabelle – chiamate CF_FRAME - che vengono successivamente elaborate. Ognuna di esse contiene informazioni che verranno successivamente copiate nella struttura CF_DATA. Infine, avviene un controllo cumulativo sull'acquisizione dei dati da parte dei quattro blocchi READ_DATA che in caso di errore setta il bit CF_DATA.COM.

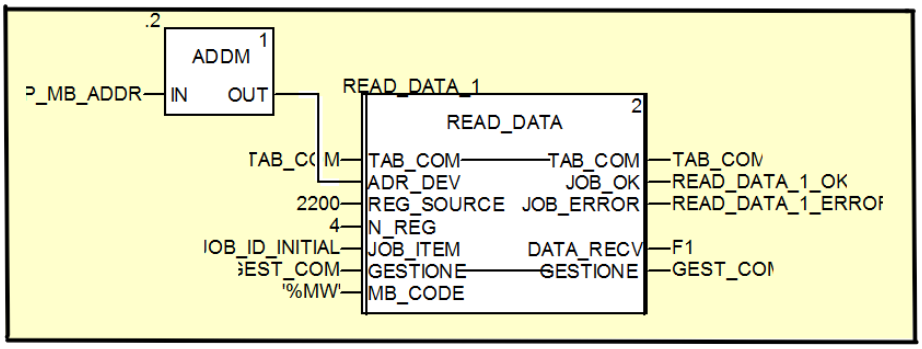


Figura 4.2.8 Blocco funzionale che richiama il DFB Type READ_DATA

- **WRITE_BIT_DATA.** Questo DFB Type è una versione estesa di una funzione integrata che permette la scrittura di dati nei registri di dispositivi esterni. I Parametri di ingresso sono l'indirizzo del dispositivo su cui scrivere ADDR_DEV, il numero di operazione assegnata JOB_ITEM, il dato da scrivere MB_EMIS (array), e il bit di disabilitazione del blocco. I parametri di uscita sono i bit di operazione conclusa positivamente/negativamente JOB_OK e JOB_ERROR, il bit di operazione in corso JOB_ON e una copia dei dati scambiati DATA_OUT. L'utilità di quest'ultimo parametro nel caso di un'operazione di scrittura è nulla, tuttavia la funzione DATA_EXCH – utilizzata come scheletro – non permette di lasciare indefinito tale campo. L'unica sezione di codice è **WRITE** che si divide in due parti principali. Nella prima si gestisce la vera e propria comunicazione; prima verificando la presenza di una richiesta di scambio e quindi tramite la funzione DATA_EXCH per effettuare la scrittura. Nella seconda parte si verifica l'esito dell'operazione e si aggiornano un certo numero di contatori e statistiche.

- **WRITE_DATA.** Questo DFB Type ha le stesse funzioni del precedente. I parametri di ingresso, oltre a quelli già descritti in WRITE_BIT_DATA, comprendono: l'indirizzo del registro da cui iniziare la scrittura REG_DEST, il numero di registri da scrivere N_REG e l'array DATA_SEND che contiene i valori che vengono trascritti, contrapposto a MB_EMIS specificato nel DFB Type precedente. I parametri di uscita coincidono invece con quelli di WRITE_BIT_DATA. Anche la sezione di codice **WRITE** condivide molte similitudini con quella già descritta. L'unica differenza sostanziale è l'utilizzo della funzione integrata WRITE_VAR come fulcro della sezione anziché DATA_EXCH.

4.3 MAST

Le sezioni sono tutte racchiuse all'interno del task principale e vengono eseguite in ordine di apparizione.

MAIN: Al contrario di altri sistemi in questo ambiente non esiste la routine principale che richiama le altre, via via che procede l'avanzamento del programma utente. In questo progetto la routine MAIN sebbene sia la prima ad essere eseguita nel ciclo macchina, viene chiamata così solo per convenzione e in questo caso contiene i comandi di avvio/arresto dei motori e il controllo del bit che segnala il blackout dell'impianto.

Un sostanziale numero di sezioni del MAST ha la sola funzione di istanziare le occorrenze dei DFB Type. Un blocco di istanza consiste sostanzialmente nell'insieme di assegnazioni degli ingressi del PLC agli input del DFB Type e la dichiarazione delle variabili di uscita su cui copiare gli output calcolati. Nello specifico:

DFB_AXxxxx. questa sezione consiste in una lista di istanze di DFB, complete di assegnazione del valore di CFG, diverso caso per caso. I DFB Type coinvolti sono:

- S_FB_Motore;
- S_FB_Inverter;
- S_FB_Sensore;
- S_FB_Deviatrice;
- S_FB_Ingresso_Ana;
- S_FB_PID;
- S_FB_Soglia;

- S_FB_Ingresso;

DFB_CEIL_HEATER. questa sezione è una lista di istanze delle occorrenze del DFB FB_Elec_Heater.

AHU_MNG. si occupa di istanziare le occorrenze del DFB Type Comandi_Stazione.

RUOTA_ENTALPICA. Questa sezione si occupa di istanziare le occorrenze del DFB Type FB_Ruota.

CONTROLLO_VAV. Questa sezione si occupa di istanziare le occorrenze del DFB Type S_FB_VAV.

CONFIGURAZIONI. Si occupa di assegnare ai parametri di ingresso dei DFB i corretti ingressi fisici o le uscite di altri DFB. In base al tipo di area servita dal PLC questa sezione può essere più o meno variegata entro uno spettro piuttosto ampio di possibilità.

ABILITAZIONI. Questa sezione ha lo scopo di abilitare o disabilitare i singoli elementi dell'equipaggiamento dell'AHU, in base alla situazione dell'ambiente in cui il PLC si trova. Tale organizzazione è necessaria per assicurare una stabilità soprattutto in fase di avvio. In ordine, la sezione esegue la seguente routine:

- I motori vengono abilitati alla marcia in caso di ESD non attivo;
- Le serrande vengono aperti quando i motori sono in marcia;
- I pressure switch vengono abilitati solo dopo 10 secondi dall'accensione dei motori, in modo che le ventole possano andare prima a regime;
- I regolatori PID vengono abilitati solo dopo 3 minuti dalla partenza dei motori;
- Le soglie vengono abilitate solamente dopo 3 minuti di marcia dei motori

Le seguenti tre sezioni hanno lo scopo di regolare il funzionamento delle reti CFCCS collegate al PLC.

ABILITAZIONI_CFCCS. Questa sezione consiste in una lista di *condizioni di attivazione*. Tali condizioni sono uno strumento messo a disposizione dall'ambiente di sviluppo per permettere allo sviluppatore di scegliere se attivare o disattivare intere sezioni di codice regolandone il valore. La lista in questione comprende 25 condizioni, chiamate MBx_ACTIVE (con x compreso tra 1 e 25), che abilitano l'esecuzione delle relative sezioni MBx_LETTURA descritte di seguito.

MBx_LETTURA. Queste 25 sezioni rappresentano ognuna una linea seriale Modbus che può essere potenzialmente collegata al PLC. Ad ognuna di esse viene affidato un indirizzo IP che coincide con quello del dispositivo EGX prescritto come client della linea. Come prima cosa il blocco crea gli indirizzi Modbus concatenando l'IP con l'indirizzo CANopen (un numero intero); per fare ciò lo sviluppatore deve specificare quanti slave (cioè i fancoil) sono collegati (fino a un massimo di 30). Infine, vengono istanziate una occorrenza di GEST_COM e una di CF_READ e vengono assegnate le variabili SCADA per le strutture CF_DATA di ogni fancoil.

MB_SCRITTURA. Questa sezione provvede ad istanziare le uniche occorrenze dei DFB Type WRITE_BIT_DATA e WRITE_DATA e ad assegnare alcuni dei parametri da associargli in ingresso. Alcuni parametri regolati da supervisore vengono utilizzati in questa sezione:

- I_LINEA indica la linea Modbus in cui deve essere propagato un messaggio e può assumere valori da 1 a 25; MB_I_FOCUS indica il nodo della linea in cui si vuole scrivere, ovvero l'indirizzo CANopen del fancoil che in questo caso può assumere valori da 1 a 30;
- TRIGGER1, … , TRIGGER20: sono 20 bit regolati da SCADA ognuno dei quali indica un comando che può essere impartito ai fancoil (vedi figura 4.3.1).

```
(* TRIGGER1 -->  ELECTRICAL HEATER MANUAL *)
(* TRIGGER2 -->  ELECTRICAL HEATER AUTOMATIC *)
(* TRIGGER3 -->  COOLING VALVE MANUAL *)
(* TRIGGER4 -->  COOLING VALVE AUTOMATIC *)
(* TRIGGER5 -->  FAN SPEED MANUAL *)
(* TRIGGER6 -->  FAN SPEED AUTOMATIC *)
(* TRIGGER7 -->  FAN COIL ON *)
(* TRIGGER8 -->  FAN COIL OFF *)
(* TRIGGER9 -->  ELECTRICAL HEATER - MANUAL REFERENCE *)
(* TRIGGER10 --> COOLING VALVE - MANUAL REFERENCE *)
(* TRIGGER11 --> FAN SPEED - MANUAL REFERENCE *)
(* TRIGGER12 --> PASSENGER SELECTION - 2 PASSENGERS *)
(* TRIGGER13 --> PASSENGER SELECTION - 4 PASSENGERS *)
(* TRIGGER14 --> ENERGY STRATEGY ENABLE *)
(* TRIGGER15 --> ENERGY STRATEGY DISABLE *)
(* TRIGGER16 --> ENERGY SAVING COOLING MODE DEAD BAND (dead band above actual setpoint) *)
(* TRIGGER17 --> ENERGY SAVING HEATING MODE DEAD BAND (dead band below actual setpoint) *)
(* TRIGGER18 --> DIFFERENCE BETWEEN CENTRAL SETPOINT LINE AND MIN SETPOINT *)
(* TRIGGER19 --> DIFFERENCE BETWEEN CENTRAL SETPOINT LINE AND MAX SETPOINT *)
(* TRIGGER20 --> SETPOINT *)
```

Figura 4.3.1 Commento con il significato dei TRIGGER in MB_SCRITTURA

È proprio quando uno di questi TRIGGER viene attivato che si attiva il codice di questa sezione. Per prima cosa viene verificata la linea a cui si vuole fare riferimento e ad essa viene abilitata la variabile che permette la scrittura. Quindi si prosegue assegnando il corretto valore

alla variabile JOB_ACT contenuta nella struttura dati di tipo DT_GESTIONE: se la scrittura implica l'utilizzo di WRITE_BIT_DATA gli viene assegnato il valore 360, se invece implica l'utilizzo di WRITE_DATA gli viene assegnato 350. Si continua specificando il nodo della rete a cui inviare il messaggio contenente il comando utilizzando la variabile MB_I_FOCUS e finalmente si specificano i dati da scrivere. Nel caso di WRITE_BIT_DATA vengono scritti sull'array MB_EMIS mentre per WRITE_DATA vengono passati all'array DATA_WR e vengono anche specificati i registri su cui scriverli. Alla fine, se l'operazione avviene con successo vedremo il valore di JOB_ACT incrementato di 1 e questo ci permetterà di riportare a 0 la variabile di abilitazione alla scrittura MB_WRITING e concludere l'esecuzione della sezione.

GESTIONE_SETPOINT. Questa sezione serve a impostare i setpoint relativi a istanze S_FB_PID che vanno a regolare valvole a 2 o 3 vie e convertitori di velocità (PIC). All'avvio del sistema (%S0 %S1 o %S13) viene assegnato a tutte le variabili IO_PC dei regolatori PID di post-riscaldamento il valore 26 che equivale al comando di funzionamento automatico. Quindi per le valvole nelle condotte che non portano ai PF viene assegnato un certo setpoint in base alla stagione mentre a quelle che servono i PF viene assegnato un setpoint cumulativo che considera anche le temperature rilevate dai PF, oppure in caso di fault di uno dei PF anche a essa viene fornito un SP di default. Per quanto riguarda i PIC l' assegnazione avviene in modo diretto. In tutte le assegnazioni viene operato il bit di comando SP della struttura K, comune a tutte le istanze viste in questo punto.

Per rendere più efficace la descrizione della prossima sezione (VALVE_FIXED_SCALING) e il prossimo capitolo sulle subroutine, risulta producente introdurre la funzione integrata SCALING. Questa funzione viene usata appunto per modificare il range di valori che una variabile numerica può assumere. I suoi parametri sono:

- IN, ovvero la variabile da scalare;
- PARA, la struttura contenente i parametri usati per scalare;
- OUT, la variabile scalata;
- STATUS, la parola di stato;

Nello specifico, la struttura PARA è composta da 4 parametri:

- in_min e in_max, il limite inferiore e superiore della scala di input;
- out_min e out_max, I limiti inferiore e superiore della scala di output;

- clip, un bit che viene posto a 1 se il valore dell'output supera i limiti imposti da out_min e out_max;

Quindi il calcolo che viene effettuato da questa funzione è:

$$OUT = (IN - in\_min) \times \frac{(out\_max - out\_min)}{(in\_max - in\_min)} + out\_min$$

Figura 4.3.2 Formula analitica della funzione integrata SCALING.

VALVE_FIXED_SCALING. Questa sezione di codice consiste in una serie di chiamate alla funzione integrata SCALING necessarie a regolare le uscite delle valvole in modo da passare della scala utilizzata internamente al PLC a quella usata per i segnali di uscita analogici.

FASCE_ORARIE: Questa sezione provvede ad istanziare un occorrenza di FB_TimeBands qualora fosse presente e in caso di modifiche da parte dell'operatore essa determina il numero di fasce attive. La seconda funzione di questo blocco di codice è la gestione delle richieste di aggiornamento dell'orario, sia su richiesta del PLC sia per l'aggiornamento del valore dei secondi e dei minuti. Si nota che lo SCADA definisce una variabile temporale per ogni AHU, di tipo intero, che viene valorizzata in minuti e aggiornata tramite counter ogni 60 aggiornamenti della timebase un secondo, RE(%S6).

Le seguenti due sezioni servono a gestire la comunicazione e i dispositivi relativi al sottosistema che comunica tramite CANopen.

DIAGNOSTICA_PF. Questa sezione gestisce lo stato del bit di errore di comunicazione sulla rete CANopen, generato quando non è possibile raggiungere il dispositivo Advantys. Per ognuno di essi viene predisposto che in caso la variabile SLAVE_ACTIV_xx sia settata a 0 allora il bit di errore di comunicazione deve essere settato a 1. Qualora il bit di attivazione torni al valore TRUE allora un delay di attivazione impostato a un secondo (TON) viene fatto partire e passato il tempo di delay l'errore di comunicazione può essere resettato. Questo meccanismo viene ripetuto per ognuno dei dispositivi Advantys presenti.

GESTIONE_CEIL_PFxxxxx. Di questa sezione esiste una versione per ogni fancoil di area pubblica (PF) presente sul progetto. Il loro scopo consiste nella gestione di ogni aspetto del funzionamento dei fancoil. Infatti, è chiaramente visibile la suddivisione in numerose parti, alcune di esse necessariamente molto simili ad alcune sezioni già descritte:

- Assegnamento del setpoint al PID di *re-heating* in fase di avviamento;
- Scaling di ingressi e uscite analogiche (utilizzando una istanza della funzione SCALING);

- Assegnazione delle variabili scada per ogni fancoil;
- Abilitazioni alla marcia, seguite da quelle per i pressure switch e delle soglie;
- Configurazione degli allarmi e degli ingressi;
- Regolazione degli ingressi dei PID;
- Impostazione dei setpoint per i PID.

Sempre relativamente alle reti di *Public Fancoil* che servono aree pubbliche, abbiamo visto come queste comprendano anche degli inverter che regolano la velocità di tali utenze. Le successive tre sezioni descrivono come viene regolato il comportamento di questi strumenti.
GESTIONE_INVERTER: questa sezione si occupa di gestire singolarmente ogni inverter. Il programma controlla per prima cosa se sono state apportate delle modifiche al comando RUN/STOP, per verificare se bisogna accendere o spegnere il dispositivo. Il sistema passa quindi a impostare il set point della velocità del motore, ma solo dopo aver effettuato diversi controlli:

- Se lo stato di Bypass è verificato si pone la velocità al massimo (10000);
- Se il livello di CO2 supera la soglia massima si abilita il funzionamento definito dalla sezione apposita CONTROLLO_LIN_FR_CO2 che regola la CO2 strategy.

Superati questi controlli il motore può assumere la velocità determinata dal controllore PID. Avviene poi un controllo che accerti che le ventole comandate dal motore non oltrepassino una soglia minima di velocità per evitare che stallino. Quindi vengono calcolati alcuni parametri relativi alla percentuale di funzionamento, si verifica il reset degli allarmi e il contatore errori CANopen. Infine, si crea un'istanza di ATV_MNG per gestire l'invio di comandi all'inverter per cui sono appena state fatte queste verifiche. Si procede in modo iterativo con tutti gli inverter gestiti nella stessa area.

CONTROLLO_LIN_FR_CO2. Il controllo lineare della velocità dell'inverter viene intrapreso quando l'apposito sensore rileva il superamento della soglia minima di CO2 misurata in parti per milione. In questa routine viene gestita la regolazione del parametro AXxxx_FR_ACTUAL_REF ovvero la velocità dell'inverter nel caso venga rilevato il superamento. Quando questo avviene è lo SCADA a comunicare al PLC che deve avviare questa procedura, attraverso la disattivazione del parametro AXxxxx_FR_DISABLED.

```
(* GESTIONE VELOCITA' INVERTER - COTROLLO LINEARE *)
(* CALCOLO IL RIFERIMENTO DI VELOCITà MA ATTUATO DALL'INVERTER SOLO SE ALLARME HIGH THRESHOLD *)
IF (AP0111_AP01111CO2.O_PC_ST < AP0111_FR_CO2_PPM_MIN) THEN
        AP0111_FR_ACTUAL_REF := AP0111_FR_PERC_MIN;
  ELSIF (AP0111_AP01111CO2.O_PC_ST > AP0111_FR_CO2_PPM_MAX) THEN
        AP0111_FR_ACTUAL_REF := AP0111_FR_PERC_MAX;
    ELSIF (AP0111_AP01111CO2.O_PC_ST >= AP0111_FR_CO2_PPM_MIN) AND (AP0111_AP01111CO2.O_PC_ST <= AP0111_FR_CO2_PPM_MAX) THEN
                (* COEFFICIENTE ANGOLARE DELLA RETTA PER DUE PUNTI*)
                COEFF_ANGOLARE := INT_TO_REAL(AP0111_FR_PERC_MAX - AP0111_FR_PERC_MIN)/INT_TO_REAL(AP0111_FR_CO2_PPM_MAX - AP0111_FR_CO2_PPM_MIN);
                (* EQUAZIONE DELLA RETTA PER DUE PUNTI*)
                AP0111_FR_ACTUAL_REF := AP0111_FR_PERC_MIN + REAL_TO_INT(COEFF_ANGOLARE*INT_TO_REAL(AP0111_AP01111CO2.O_PC_ST - AP0111_FR_CO2_PPM_MIN));
        ELSE
        AP0111_FR_ACTUAL_REF := AP0111_FR_PERC_MIN;
END_IF;

(* Non sono concesse valori di VELOCITA' superiori al 100% o inferiori a 0 *)
(* Dovuti ad errori di conversione int real - real int *)
IF AP0111_FR_ACTUAL_REF < 0 THEN
        AP0111_FR_ACTUAL_REF := 0;
END_IF;
IF AP0111_FR_ACTUAL_REF > 10000 THEN
        AP0111_FR_ACTUAL_REF := 10000;
END_IF;
```

Figura 4.3.3 Controllo lineare della velocità dell'inverter in funzione del livello di CO2 rilevato nell'ambiente [AP0111]

DIAGNOSTICA_INVERTER. Per ogni inverter vengono impostati dei delay avviati in fase di spegnimento e accensione per settare o resettare rispettivamente il bit di errore di comunicazione. La variabile che abilita questi timer è CAN_DIAG.SLAVE_ACTIV_X ovvero un parametro di un array (CAN_DIAG) di tipo T_COM_BMX_EXPERT. Questo tipo di array è un IODDT specifico del CANopen. In questa sezione viene dichiarato il delay di accensione (quello di spegnimento dichiarato nella sezione GESTIONE_INVERTER) e si controlla il segnale ERR_COMM che deve essere comunicato allo SCADA.

EMERGENCY_MNG. Questa sezione non è presente in ogni progetto ma solo in quelle previste aventi un ruolo nelle procedure di emergenza; nello specifico si tratta di quelle aree che comprendono scale o le aree pubbliche individuate per raggruppare i passeggeri in caso di pericolo. La sua funzione principale è la gestione del compressore di emergenza che viene acceso in caso di blackout per assicurare il ricambio dell'aria minimo necessario. La gestione avviene tramite la lettura degli ingressi provenienti dal compressore stesso, che segnalano lo stato di attivazione e quello di allarme. Il PLC in base al valore di questi regola l'uscita che aziona il *raffreddatore d'emergenza*. Vengono monitorate anche le temperature rilevate da tutti i trasmettitori, in modo da considerare sempre quella più alta. La sezione permette di selezionare tramite un selettore locale tra due modalità di funzionamento: principale e di emergenza.

```
(*STATUS WORD*)
(* .0 Run*)
(* .1 Fault*)
(* .2 Chiller Comandato *)
(* .3 Manuale/automatico stato *)
(* .4 Comando Manuale *)
(* .5 in Manuale *)
(* .6 reset allarme compressore *)
```

Figura 4.3.4 Status word del compressore di emergenza.

4.4 Safety Management Control System (SMCS)

Come anticipato, il sistema è provvisto di un PLC che non viene associato a nessuna unità di trattamento dell'aria ma ha la funzione di monitorare lo stato di emergenza di tutto l'impianto e intraprendere azioni per la gestione delle emergenze.

La struttura HW di tale PLC comprende la CPU utilizzata anche in tutti gli altri PLC dell'impianto e un modulo di comunicazione Ethernet che si occupa di comunicare con tutti i PLC che devono attuare la smoke strategy. Ogni dispositivo figura come slave della rete di cui il SMCS è il master. Per attuare questo comportamento si fa in modo che essi ricevano una command word sul proprio registro %MW1000 e che scrivano un feedback sul proprio registro %MW1001. Questi registri saranno effettivamente collegati come ingressi e uscite fisiche del SMCS.

	IP address	Device Name	Unit ID	Slave Syntax	Health Timeout (ms)	Repetitive rate (ms)	RD Master Object	RD Ref Slave	RD length	Last value (Input)	WR Master Object	WR Ref Slave	WR length	Description
1	192.168.1.1		255	IEC 0	1500	250	%MW1000	%MW1001	1	Set to 0	%MW2000	%MW1000	1	AP0111
2	192.168.1.6		255	IEC 0	1500	250	%MW1001	%MW1001	1	Set to 0	%MW2001	%MW1000	1	AP0411
3	192.168.1.8		255	IEC 0	1500	250	%MW1002	%MW1001	1	Set to 0	%MW2002	%MW1000	1	AT0411
4	192.168.1.7		255	IEC 0	1500	250	%MW1003	%MW1001	1	Set to 0	%MW2003	%MW1000	1	AC0411
5	192.168.2.7		255	IEC 0	1500	250	%MW1004	%MW1001	1	Set to 0	%MW2004	%MW1000	1	AC0621
6	192.168.2.11		255	IEC 0	1500	250	%MW1005	%MW1001	1	Set to 0	%MW2005	%MW1000	1	AT1121
7	192.168.3.2		255	IEC 0	1500	250	%MW1006	%MW1001	1	Set to 0	%MW2006	%MW1000	1	AC0131
8	192.168.3.1		255	IEC 0	1500	250	%MW1007	%MW1001	1	Set to 0	%MW2007	%MW1000	1	AP0132
9	192.168.3.4		255	IEC 0	1500	250	%MW1008	%MW1001	1	Set to 0	%MW2008	%MW1000	1	AP0631
10	192.168.3.5		255	IEC 0	1500	250	%MW1009	%MW1001	1	Set to 0	%MW2009	%MW1000	1	AP0531
11	192.168.3.6		255	IEC 0	1500	250	%MW1010	%MW1001	1	Set to 0	%MW2010	%MW1000	1	AC0431
12	192.168.3.7		255	IEC 0	1500	250	%MW1011	%MW1001	1	Set to 0	%MW2011	%MW1000	1	AC0831
13	192.168.3.12		255	IEC 0	1500	250	%MW1012	%MW1001	1	Set to 0	%MW2012	%MW1000	1	AT1131
14	192.168.4.1		255	IEC 0	1500	250	%MW1013	%MW1001	1	Set to 0	%MW2013	%MW1000	1	AC0142
15	192.168.4.4		255	IEC 0	1500	250	%MW1014	%MW1001	1	Set to 0	%MW2014	%MW1000	1	AC0241
16	192.168.4.6		255	IEC 0	1500	250	%MW1015	%MW1001	1	Set to 0	%MW2015	%MW1000	1	AC0242
17	192.168.4.5		255	IEC 0	1500	250	%MW1016	%MW1001	1	Set to 0	%MW2016	%MW1000	1	AP0441
18	192.168.4.7		255	IEC 0	1500	250	%MW1017	%MW1001	1	Set to 0	%MW2017	%MW1000	1	AC0541
19	192.168.4.10		255	IEC 0	1500	250	%MW1018	%MW1001	1	Set to 0	%MW2018	%MW1000	1	AT1141
20	192.168.5.1		255	IEC 0	1500	250	%MW1019	%MW1001	1	Set to 0	%MW2019	%MW1000	1	AP0151
21	192.168.5.2		255	IEC 0	1500	250	%MW1020	%MW1001	1	Set to 0	%MW2020	%MW1000	1	AC0151
22	192.168.5.6		255	IEC 0	1500	250	%MW1021	%MW1001	1	Set to 0	%MW2021	%MW1000	1	AP0451
23	192.168.5.5		255	IEC 0	1500	250	%MW1022	%MW1001	1	Set to 0	%MW2022	%MW1000	1	AC0451
24	192.168.5.7		255	IEC 0	1500	250	%MW1023	%MW1001	1	Set to 0	%MW2023	%MW1000	1	AC0651
25	192.168.5.11		255	IEC 0	1500	250	%MW1024	%MW1001	1	Set to 0	%MW2024	%MW1000	1	AT1451
26	192.168.6.1		255	IEC 0	1500	250	%MW1025	%MW1001	1	Set to 0	%MW2025	%MW1000	1	AC0161
27	192.168.6.3		255	IEC 0	1500	250	%MW1026	%MW1001	1	Set to 0	%MW2026	%MW1000	1	AP0261
28	192.168.6.4		255	IEC 0	1500	250	%MW1027	%MW1001	1	Set to 0	%MW2027	%MW1000	1	AP0561
29	192.168.6.7		255	IEC 0	1500	250	%MW1028	%MW1001	1	Set to 0	%MW2028	%MW1000	1	AC0661
30	192.168.6.10		255	IEC 0	1500	250	%MW1029	%MW1001	1	Set to 0	%MW2029	%MW1000	1	AC1061
31	192.168.6.13		255	IEC 0	1500	250	%MW1030	%MW1001	1	Set to 0	%MW2030	%MW1000	1	AT1461
32	192.168.6.12		255	IEC 0	1500	250	%MW1031	%MW1001	1	Set to 0	%MW2031	%MW1000	1	AT0361
33	192.168.0.10		255	IEC 0	10000	10000	%MW1032	%MW1050	1	Hold last	%MW2032	%MW999	33	IAMCS
34	192.168.3.8		255	IEC 0	1500	250	%MW1033	%MW1001	1	Set to 0	%MW2065	%MW1000	1	AP1031
35	192.168.0.10		255	IEC 0	10000	10000	%MW1034	%MW799	0	Hold last	%MW2066	%MW1099	26	IAMCS_variante

Figura 1 sezione riassuntiva IO Scanning contenente la lista dei dispositivi collegati e dei loro registri dove il sistema può operare.

Il suo MAST è formato da varie sezioni.

DFB_SMCS. Si occupa di istanziare l'attivazione di ogni tipo di strategia, ognuna delle quali coinvolge aree separate della nave. Per fare ciò viene utilizzato il DFB Type S_FB_Ingresso, analizzato nel paragrafo 4.2.

INPUT_ASSIGNMENT. In questa sezione vengono assegnate le smoke strategy da SMCS leggendo la WORD SMCS_INPUT_STRATEGY immagazzinata nella memoria di sistema all'indirizzo %MW500. Ogni bit di tale WORD corrisponde al parametro I_ST che viene passato in ingresso alle istanze dichiarate in DFB_SMCS. In caso di sistema SMCS considerato "morto" le strategie lette non saranno più attendibili e le variabili di ingresso manterranno il valore registrato per ultimo, prima del distacco. La funzione principale di questa sezione è la regolazione del bit di attivazione nella parola di comando. Tale parola di comando è quella che l'SMCS comunica ai singoli PLC e viene denominata SMCS_COMMAND_AXxxxx. Il bit meno significativo di tale parola contiene il bit di attivazione. Si nota che per alcune aree della nave, l'attivazione della strategia può dipendere da più di una istanza.

PLC_TO_SCADA. Questa sezione si occupa di assegnare alle relative variabili SCADA il valore del feedback registrato proveniente dai PLC. Tali valori sono immagazzinati nei registri %MW dal 1000 al 1033.

OUTPUT_COMULATIVO. Questa sezione tiene traccia delle variazioni che avvengono nelle word di feedback e fornisce un report cumulativo per ognuna delle istanze dichiarate in DFB_SMCS. Opera sui bit della parola di feedback che indicano rispettivamente:

- 0: bit di attivazione della smoke strategy [OR];
- 1: smoke strategy eseguita correttamente [AND];
- 2: allarmi delle stazioni in condizione di smoke strategy [OR];
- 3: allarme di posizionamento delle serrande [OR];
- 4: posizionamento delle serrande corretto [AND];
- 15: stato del bit di KEEP_ALIVE [AND].

Bit intermedi non vengono utilizzati per il feedback cumulativo.

IAMCS. Questa sezione si occupa di aggiornare e comunicare lo stato di ogni elemento che deve essere monitorato anche dal sistema IAMCS, ovvero lo stato di RUN di ogni ventola e il loro eventuale stato di guasto e altri parametri. Il sistema IAMCS consiste in un sistema indipendente che si occupa del controllo degli allarmi dell'intera nave. L'interfaccia con questo sistema consiste in un pannello diverso dal supervisore del sistema HVAC. Esso rimane fisso sulla stessa pagina per monitorare costantemente l'andamento degli allarmi.

KEEP_ALIVE. Questa sezione racchiude una serie di funzioni utili a stabilire se le informazioni provenienti dai vari dispositivi della rete possono ritenersi attendibili o meno. Di fatto si tratta di contatori basati su timebase %S6 che rilevano il cambiamento di alcune variabili associate a ingressi fisici provenienti dai dispositivi da controllare, ovvero SMCS e IAMCS. Entrambi vengono considerati "morti" se per più di 180 secondi le variabili

SMCS_KEEP_ALIVE_COUNTER e IAMCS_KEEP_ALIVE mantengono lo stesso valore. Una loro modifica resetta invece i counter ad esse associati.

Diag_ETH. Questa sezione realizza la lettura dello stato dell'interfaccia Ethernet. Basandosi su timebase %S7 (60 secondi), viene richiamata una funzione integrata read_sts() che è usata per leggere le *parole di stato* di un modulo o un interfaccia integrata che contengono le loro modalità operative. Il parametro della funzione in questo caso è di tipo T_COM_ETH_BMX e leggendo i suoi parametri si può determinare se sono stati registrati errori: durante la lettura dello stato del canale, l'aggiustamento del canale, l'invio di un comando sul canale, se il canale è inattivo o se l'applicazione presenta qualche fault.

5. Monitoraggio dei consumi energetici

5.1 Obiettivi della modifica e direttive di sviluppo

In seguito alla consegna del lavoro terminato e descritto al capitolo 4, il cliente ha ritenuto necessario apporre una piccola modifica al progetto aggiungendo una sezione che monitori il consumo di tutti i motori coinvolti dal sistema HVAC.

Questa applicazione richiede lo sviluppo su Unity Pro XL del codice che sarà utilizzato dall'HMI per comunicare ai motori le istruzioni provenienti dal PLC e per mostrare sul pannello operatore le seguenti informazioni:

- Se il motore/inverter sta funzionando in modalità Bypass;
- Se il motore/inverter sta funzionando con il fault KW_FAULT attivo;
- La potenza assorbita in kW;
- La potenza nominale del motore.

I dati forniti dal cliente comprendono una lista di Tag, ovvero di variabili, complete di specifica sul tipo, un breve commento sul loro significato, il range di valori che possono assumere, il formato con il numero di cifre decimali da considerare e infine le unità di misura. Esse rappresentano alcune delle informazioni necessarie per l'esecuzione del calcolo e per evidenziare il più possibile il rispetto delle richieste del cliente vengono raggruppate in una struttura dati chiamata ENERGY_STRUCT posizionata nella sezione Inputs/Outputs del DFB. Infine, viene incluso un diagramma di flusso dettagliato che sarà utilizzato anche come base di partenza per progettare le pagine dell'HMI.

I motori che vengono utilizzati in questo progetto sono di tre tipologie:

- Motori elettrici a singola fase a una velocità;
- Motori elettrici a singola fase a due velocità;
- Motori elettrici con VFD (Variable Frequency Drive – Variatore di frequenza) ovvero un sistema usato per regolare la velocità di motori a corrente alternata.

In aggiunta a questi vi è una cospicua presenza di motori alimentati a corrente continua che vengono usati come attuatori per regolare elementi minori della rete come i damper; tuttavia, non vengono considerati visto il loro tempo di attività che risulta marginale.

Chiaramente l'elemento comune alle tre categorie elencate è il motore a corrente alternata. Questo apparecchio è formato da *statore* e *rotore,* i quali generano un campo magnetico tramite l'utilizzo di magneti permanenti o avvolgimenti elettrici di cavi attorno ad un nucleo di lamierini di ferro (elettromagnete).

I motori monofase sono di tipo asincrono e alimentati a 220V, caratterizzati cioè da una velocità angolare del rotore inferiore alla velocità di rotazione del campo magnetico generato dagli avvolgimenti dello statore. In virtù del suo funzionamento viene anche detto motore a induzione. Per agevolare l'accensione di questi dispositivi viene utilizzato un avviatore statico ATS01 di Schneider Electric che permette di fornire un avviamento agevole atto a prolungare la vita utile dei dispositivi serviti e per limitare il rumore nelle fasi di avvio e di arresto. Inoltre, consente di controllare precisamente il tempo di avviamento per far fronte all'inerzia del sistema e ai suoi limiti meccanici.

I motori a cui vengono collegati i variatori di velocità sono invece motori trifase, il cui funzionamento è basato sull'applicazione del principio del campo magnetico rotante ad un insieme trifase di correnti in ingresso ovvero un sistema di correnti i cui fasori sono sfasati tra loro di 120 gradi. I variatori di velocità, indicati come VFD (variable frequency drive) sono dispositivi usati per regolare la velocità di rotazione dei motori a corrente alternata. Hanno la capacità di regolare la frequenza o il voltaggio – grandezze dipendenti l'una dall'altra - e fornire la combinazione che definisce univocamente la velocità di rotazione desiderata. Come si può vedere in figura 5.1.1 il segnale fornito in ingresso a questi dispositivi è prelevato direttamente dal sistema trifase di alimentazione principale, ovvero un segnale sinusoidale a 690V/60Hz, che però viene limitato a 480V come richiesto dalle specifiche della macchina. All'interno, la struttura del dispositivo si può sintetizzare come la serie di un raddrizzatore seguito da un inverter:

- Il raddrizzatore permette di ridurre la frequenza fino a trasformarlo in un segnale continuo;
- L'inverter è un dispositivo che prende in ingresso una corrente continua e fornisce in uscita un segnale sinusoidale della frequenza desiderata diversa da zero, in questo caso un segnale con una frequenza che può assumere valori da 0 a 500Hz.

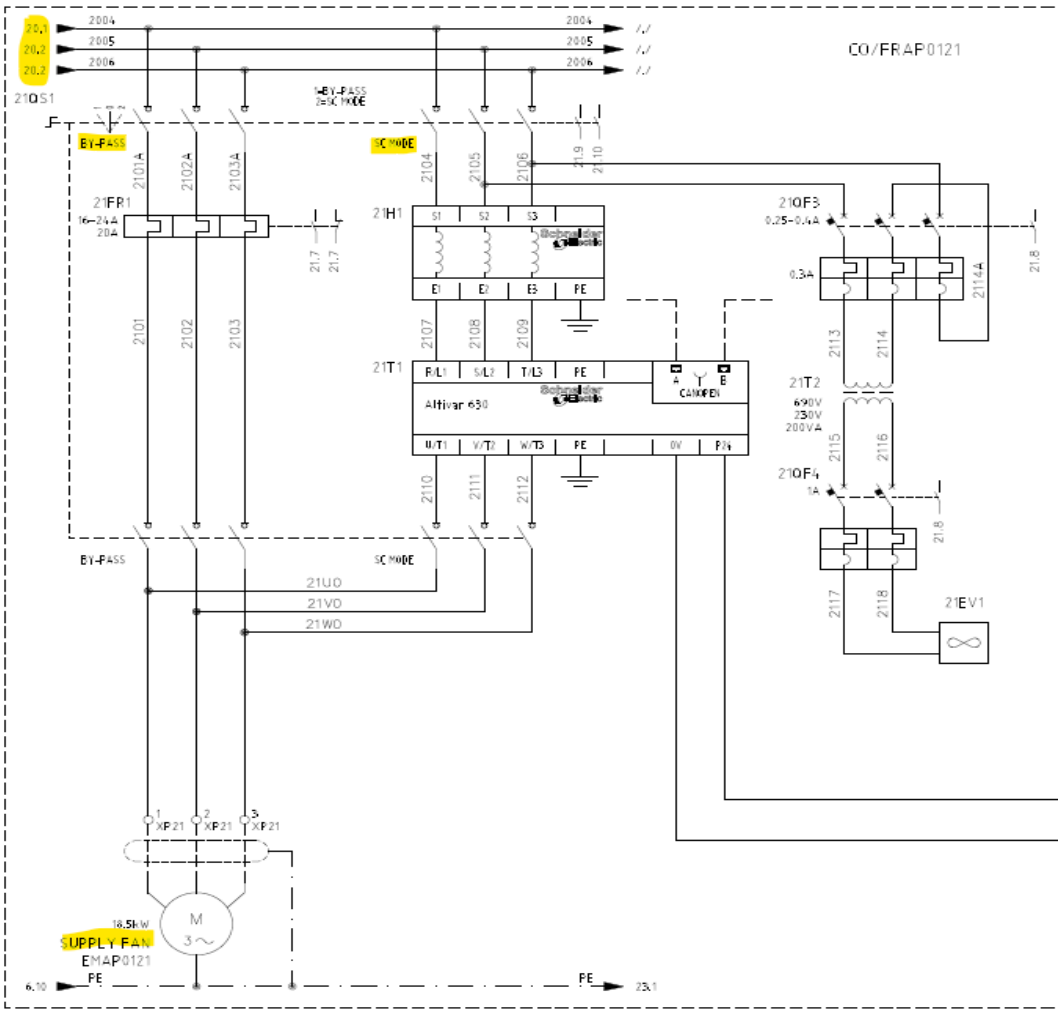


Figura 5.1.1 dettaglio di uno schema elettrico, motore elettrico servito da VFD Altivar630

5.2 Sviluppo del codice

La modifica consiste nell'aggiunta di un DFB Type, in questo caso chiamato ENERGY_MNG, composto di una sola sezione che racchiuderà tutti i calcoli svolti dal software. Il DFB Type sarà successivamente istanziato a dovere per ogni motore.

Il procedimento seguito rispecchia minuziosamente i dettagli esposti nel diagramma di flusso mostrato in figura 5.2.1.

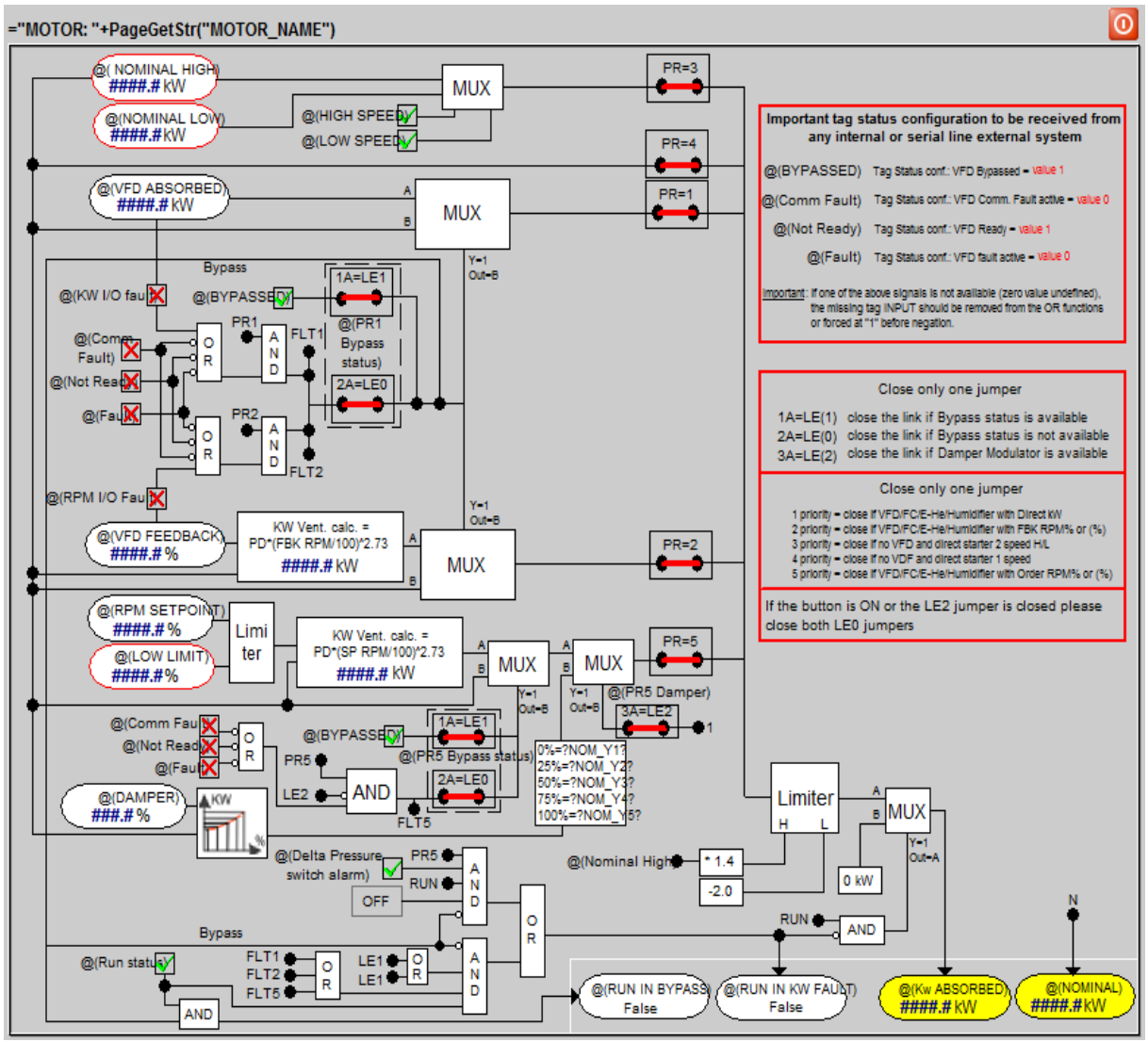


Figura 5.2.1 Logica combinatoria per la misura del consumo energetico

5.2.1 Calcolo della potenza assorbita da un motore

Il sistema, anzitutto distingue la tipologia di motore per cui è richiesto il calcolo della potenza assorbita. Nel sistema sono presenti due tipologie di motore, motori a una fase e inverter; a loro volta i motori a una fase sono suddivisi in motori a velocità singola e motori a velocità doppia. In base a quale delle tre opzioni viene selezionata il programma seguirà il ramo corrispondente. Oggettivamente alla fine risulteranno cinque rami diversi anziché tre, in quanto il consumo degli inverter può essere attribuito in tre modi differenti. In questo scenario, ogni ramo viene denominato "priorità" (PR):

- [PR = 1] si ha quando il motore è del tipo inverter ed è disponibile il valore della potenza assorbita in unità ingegneristiche (kW). La logica combinatoria comprende un multiplexer che legge in ingresso il valore di potenza assorbita (A) e il valore nominale del motore (B). L'uscita è invece controllata un solo bit di selezione che se

viene settato fa propagare l'ingresso B. Questo bit proviene da un altro segmento di logica combinatoria che considera se è disponibile e attivo lo stato di Bypass oppure se è stato attivato qualche fault;

- [PR = 2] per quando si ha un motore di tipo inverter e il valore del consumo rilevato è calcolato sulla base del feedback (FBK) fornito dal motore e misurato in % rispetto al totale della potenza erogabile. La logica combinatoria prevede un multiplexer con due ingressi. L'ingresso B corrisponde al valore nominale (PD) dell'inverter e viene selezionato se il bit di selezione, lo stesso del caso [PR = 1] è attivo. L'ingresso A corrisponde invece a un valore che viene calcolato secondo la formula (i) e viene selezionato quando il bit di selezione è spento, cioè quando l'impianto funziona normalmente;

$$KW = PD * \left(\frac{FBK}{100}\right)^{2.73} \qquad \text{(i)}$$

- [PR = 3] in questo caso il motore è monofase a due velocità e il valore viene fornito direttamente in kW. La logica combinatoria comprende un multiplexer che prende in ingresso il valore nominale della potenza erogata massima e minima; l'uscita viene selezionata da due bit di controllo @(HIGH SPEED) e @(LOW SPEED), uno dei quali viene settato a uno direttamente da pannello;
- [PR = 4] quando il motore è monofase e viene fornito direttamente il valore nominale in kW. La logica combinatoria in questo caso è inesistente a monte del jumper che seleziona la priorità;
- [PR = 5] il motore è di tipo inverter e il valore del consumo viene fornito da un set-point di percentuale di apertura dei damper impostato dall'utente. La logica combinatoria in questo caso è composta da quattro step risultando la più complessa. Per prima cosa avviene una verifica che il set-point impostato dall'utilizzatore non sia minore di una certa soglia che potrebbe comportare lo stallo delle ventole. Successivamente subisce la stessa trasformazione (i) vista al punto [PR = 2]. Prosegue passando attraverso un multiplexer con la stessa logica di funzionamento visto al punto [PR = 1], che in caso di fault o funzionamento in stato di Bypass seleziona in uscita il valore nominale dell'inverter. Infine, passa attraverso un ulteriore multiplexer il cui bit di selezione combacia con la disponibilità della modalità Damper Modulator. Questa modalità permette di regolare la potenza dei motori fissando la percentuale di apertura dei damper nei condotti di mandata. A tale scopo viene definita una funzione che ritorna il valore corretto da considerare (vedi paragrafo 5.2.3).

In talune situazioni, descritte in precedenza, si vede il sistema operare in condizione di "fault". Questa condizione non implica necessariamente l'impossibilità per il motore di operare ma piuttosto mostra l'esistenza di qualche anomalia che non permette al PLC di ricevere una stima attendibile del consumo di energia. I fault specificati dal programma comprendono:

- **KW I/O fault**, cioè quando il valore calcolato e quello reale di potenza assorbita non combaciano;
- **Comm fault**, quando c'è un guasto sulla rete di comunicazione tra il motore e il PLC;
- **Not ready**, quando il motore non è nello stato READY, condizione necessaria affinché il motore possa essere azionato;
- **Fault**, indicato come VFD_FAULT indica un effettivo guasto dell'inverter.
- **RPM I/O Fault**, similmente a KW_IO_FAULT indica una discrepanza tra il valore dichiarato e quello effettivo.

Un'altra situazione rilevante è rappresentata dal funzionamento in modalità Bypass. Questa funzione è selezionata manualmente dall'operatore sul pannello di controllo dell'inverter e viene utilizzata quando per qualsiasi motivo – spesso in caso di guasto - si vuole isolare il motore dall'inverter, separandolo quindi dalla funzione di regolazione della velocità che lo caratterizza e lo distingue da un normale motore elettrico. Così facendo, per quanto riguarda la potenza assorbita, non resta che considerare quella nominale. È bene notare che la scelta di operare un motore in questa modalità permette di non privare un ambiente del condizionamento dell'aria necessaria in caso di guasto dell'inverter, e contemporaneamente viene notificata al software di supervisione così da monitorare e gestire in modo efficiente la problematica.

Una volta che la priorità è stata definita e la modalità di funzionamento è stata determinata al netto di fault, bypass e damper modulator, il valore passa attraverso un limitatore che ha la funzione di limitare errori di calcolo del PLC che portano a risultati impossibili (si può pensare a questa evenienza come ad una divisione per un valore molto vicino allo 0) e infine attraverso un multiplexer che verifica vari dettagli spiegati più estensivamente nel capitolo 5.2.3, tra cui lo stato di accensione dell'impianto (tasto ON/OFF). Se le verifiche hanno esito negativo verrà visualizzato a schermo il valore 0 kW altrimenti visualizzeremo finalmente il valore di potenza assorbita dal motore in questione.

5.2.2 Parametri di ingresso e di uscita del DFB Type

Procederemo ora a esaminare le variabili prese in considerazione dal DFB Type.

Viene utilizzata la sezione di input, in cui sono specificati i parametri definiti in altre parti del programma che vengono consultati all'interno di questo FB, e la parte di Inputs/Outputs in cui viene dichiarata la struttura ENERGY_STRUCT, la quale contiene tutte le variabili che sono state espressamente indicate nel documento redatto dal cliente, in modo da rendere la lettura il più coerente possibile con le richieste.

Gli input comprendono un insieme di valori comunicato in tempo reale dal dispositivo controllato:

- VFD_POWER, la potenza assorbita dall'inverter;
- VFD_RPM_FBK, la percentuale di utilizzo dell'inverter in priorità 5;
- MOD_DMP_FBK, percentuale di modulazione dei damper in priorità 5;
- RUN, bit di stato che segnala il funzionamento;
- VFD_COMM_FAULT, guasto di comunicazione;
- KW_IO_FAULT;
- RPM_IO_FAULT;
- VFD_FAULT;
- VFD_READY;
- VFD_BYPASSED;
- LOW_SPD, motore impostato a velocità bassa;
- HIGH_SPD, motore impostato a velocità alta;
- PRES_SW_ALM, allarme collegato ai sensori di pressione nelle ventole.

Il PLC legge questi valori, oltre agli input contenuti nella struttura ENERGY_STRUCT, ovvero il valore della potenza nominale erogata dal motore (Massima e minima in caso di motore a fase singola a due velocità) e lo stato della command word (CMDW) che contiene le informazioni che il PLC comunica all'inverter.

La CMDW è una WORD che contiene dati nei primi 9 bit, lasciando i restanti vuoti.

CMDW.0	Motore in priorità 1
CMDW.1	Motore in priorità 2
CMDW.2	Motore in priorità 3
CMDW.3	Motore in priorità 4

CMDW.4	Motore in priorità 5
CMDW.5	PR1, stato di Bypass (1 = LE1; 0 = LE0)
CMDW.6	PR5, stato di Bypass (1 = LE1; 0 = LE0)
CMDW.7	PR5, LE2 Dumper Modulator disponibile
CMDW.8	PR5, pulsante di accensione

Il programma riesce facilmente dai primi 5 bit a derivare a quale categoria appartiene il motore: inverter, motore a una fase con una velocità, motore a una fase con due velocità. In base a questa e alle informazioni acquisite dalla CMDW, il PLC avrà così determinato il percorso da seguire nello sviluppo dei calcoli necessari a determinare la potenza assorbita e la Status word (STW).

Le restanti variabili della ENERGY_STRUCT sono gli output:

- VFD_ABS, la potenza assorbita dall'inverter;
- ABS, la potenza assorbita;
- ABS_RATE, il rate di potenza assorbita;
- VFD_RPM_FBK, la velocità dell'inverter (a differenza dell'omonimo parametro di ingresso, il dato è convertito e scalato mantenendo comunque lo stesso significato)
- VFD_PR2_CALC, la potenza assorbita quando l'inverter è in priorità 2;
- VFD_RPM_SET, setpoint dell'inverter.

Infine, viene definita la status word:

STW.0	Tipo di motore: Diretto (0) o Inverter (1)
STW.1	Motore diretto a due velocità: massima (0), minima (1)
STW.2	Inverter: Bypass attivato
STW.3	Motore in funzione
STW.4 to STW.7	Fault: I/O, comunicazione, (NOT) ready, VDF
STW.8	Funzionamento in Bypass
STW.9	Fault: RPM I/O
STW.10 to STW.12	FLT1, FLT2, FLT5 – tipo di fault (uno solo attivato)
STW.13	Allarme dei sensori di differenza di pressione (stallo ventole)
STW.14	Funzionamento in KW fault
STW.15	Bypass richiesto (variabile intermedia) o effettivo (0 virtual, 1 real)

5.2.3 Implementazione del codice

Ora che abbiamo visto la logica combinatoria che regola questo processo di calcolo e conosciamo tutte le variabili che vengono coinvolte, possiamo finalmente illustrare come viene implementato il codice nella unica sezione del DFB Type ENERGY_MNG.

Come prima cosa il sistema legge il valore nominale dichiarato dal motore e tutti i bit della CMDW, associandoli a delle variabili interne; dai primi cinque bit della WORD è immediato ricavare il tipo di motore impiegato (inverter o a singola fase), usando un semplice IF. Successivamente, vengono assegnati i bit della STW (16 bit rilevanti); alcuni bit vengono inizializzati a parametri di ingresso e altri a variabili private. Proseguendo, vengono convertiti e scalati alcuni parametri di ingresso espressi in unità ingegneristiche o in percentuali, in modo da poterle utilizzare con la scala più consona. Nello specifico:

- la variabile VFD_RPM_FBK viene convertita da intero a reale e viene scalata di un fattore 100, trasformandolo quindi in un valore percentuale. In questo modo può essere utilizzata nella formula (i) del paragrafo 5.2.1;
- VFD_RPM_CMD, è il valore di funzionamento percentuale dell'inverter quando è in priorità 5, cioè quando viene impostato un setpoint dall'utilizzatore. Segue lo stesso trattamento e viene utilizzato per essere confrontato con il valore di soglia per evitare lo stallo delle ventole;
- VFD_POWER è il valore di potenza assorbita fornito dal motore stesso. A differenza degli altri valori (%) questo è un valore espresso in kW ma subisce lo stesso procedimento degli altri;
- MOD_DMP_FBK è la percentuale di apertura dei damper. Viene convertita da intero a reale e viene scalata di un fattore cento per poterla utilizzare all'interno di una funzione che verrà spiegata successivamente in questo paragrafo.

Quindi, a prescindere dall'utilizzo interno alla routine di questi parametri, il motivo per cui subiscono queste modifiche è per riportarli dalla loro scala personalizzata a quella standard per poi copiare il valore corretto nelle variabili di ENERGY_STRUCT.

Si prosegue con la definizione di alcune variabili di appoggio di tipo bool che segnalano stati di fault: app_flt, che è verificata quando è presente un fault di comunicazione, VFD_FAULT o il fault VFD_READY (negato); FLT1 che è attivato solo se è presente un fault e il motore è in priorità 1; FLT2 se in priorità 2; FLT5 se in priorità 5 e se la modulazione damper non è disponibile.

Si verifica quindi se l'inverter sta venendo bypassato. In questi casi il valore registrato viene fatto combaciare con il valore nominale del motore: in priorità 1, 2 e 5 si considera l'inverter bypassato se è impostato manualmente il Bypass oppure se è presente un fault di qualsiasi

tipo. Questo controllo ci permette di pilotare il bit di selezione e quindi le uscite del primo multiplexer posto a monte del ponticello di priorità permettendoci di propagare il valore corretto da registrare in ognuno dei cinque casi.

Per le priorità 1, 3, 4 sono sufficienti dei semplici IF; vale invece la pena approfondire il processo che avviene nelle restanti casistiche.

In caso di priorità 2 prima di controllare il multiplexer è reso necessario un controllo sul segno della variabile di appoggio app_VFD_RPM_FBK – la versione convertita e scalata di VFD_RPM_FBK vista in precedenza – e in caso affermativo significa che le ventole alle quali il motore è collegato stanno ruotando in senso contrario a quello previsto, indice del fatto che il motore è spento e le ventole sono soggette ai casuali spostamenti di aria all'interno delle condutture, e il valore viene riportato a 0.

Successivamente si applica la formula sperimentale (i) di cui l'azienda non conosce i particolari ma che si presume sia collegato alla considerazione della viscosità dell'aria e all'attrito misurato generato dalle parti meccaniche del motore. Una volta svolti questi calcoli si procede a verificare lo stato del multiplexer e a propagare il valore di potenza assorbita. Il processo appena descritto è riportato in figura 5.2.1.

```
(* motor in Priority 2 : VFD/FC/E-He/Humidifier/ with FBK RPM% or (%) *)
IF PR2 THEN

        IF app_VFD_RPM_FBK < 0.0 THEN
                app_VFD_RPM_FBK := 0.0;
        END_IF;

        KW_Absorbed := NOM_H * (app_VFD_RPM_FBK / 100.0) ** 2.73;

        ENERGY_STRUCT.VFD_PR2_CALC := KW_Absorbed;      (* Absorbed power from VFD in priority 2 *)

        IF app_MUX_PR2 THEN                             (* Running in Bypass *)

                KW_Absorbed := NOM_H;

        END_IF;

ELSE

        ENERGY_STRUCT.VFD_PR2_CALC := 0.0;              (* Absorbed power from VFD in priority 2 *)

END_IF;
```

Figura 5.2.1 Codice per la priorità 2

Per quanto riguarda la priorità cinque si verifica prima di tutto che il motore non scenda sotto la soglia operativa che potrebbe portare allo stallo delle ventole, nel qual caso si aumenta la sua potenza fino al valore impostato. Si procede quindi ad applicare la formula (i) e alla verifica dello stato del bit di selezione del multiplexer. Questo ramo del programma prevede un passaggio aggiuntivo necessario per determinare il giusto valore da propagare: il multiplexer che è regolato dall'attivazione della modalità "damper modulation" accennata precedentemente.

Questa funzione definisce la potenza nominale del motore in funzione della percentuale di apertura dei damper. Individua con una serie di IF mostrati in figura 5.2.2 il range a cui appartiene il valore specificato nella variabile app_MOD_DMP_FBK e quindi procede a calcolare il coefficiente angolare della funzione in quel segmento e ad applicare la formula che determina la quantità di potenza assorbita.

```
IF app_MUX_PR5_2 THEN                          (* Modulating damper *)

        NOM_X1 := 0.0;
        NOM_X2 := 25.0;
        NOM_X3 := 50.0;
        NOM_X4 := 75.0;
        NOM_X5 := 100.0;

        IF app_MOD_DMP_FBK < NOM_X2 THEN
                InputLow   := NOM_X1;
                InputHigh  := NOM_X2;
                EValueLow  := ENERGY_STRUCT.NOM_Y1;
                EValueHigh := ENERGY_STRUCT.NOM_Y2;

        ELSIF app_MOD_DMP_FBK >= NOM_X2 AND app_MOD_DMP_FBK < NOM_X3 THEN
                InputLow   := NOM_X2;
                InputHigh  := NOM_X3;
                EValueLow  := ENERGY_STRUCT.NOM_Y2;
                EValueHigh := ENERGY_STRUCT.NOM_Y3;

        ELSIF app_MOD_DMP_FBK >= NOM_X3 AND app_MOD_DMP_FBK < NOM_X4 THEN
                InputLow   := NOM_X3;
                InputHigh  := NOM_X4;
                EValueLow  := ENERGY_STRUCT.NOM_Y3;
                EValueHigh := ENERGY_STRUCT.NOM_Y4;

        ELSIF app_MOD_DMP_FBK >= NOM_X4 THEN
                InputLow   := NOM_X4;
                InputHigh  := NOM_X5;
                EValueLow  := ENERGY_STRUCT.NOM_Y4;
                EValueHigh := ENERGY_STRUCT.NOM_Y5;

        END_IF;

        Ratio:=(EValueHigh-EValueLow)/(InputHigh-InputLow);

        KW_Absorbed:=(EValueLow + (( app_MOD_DMP_FBK - InputLow) * Ratio )) / 100.0 * NOM_H;
```

Figura 5.2.2 Implementazione della funzione Damper Modulation

Per finire si verifica che il risultato di questa operazione sia contenuto all'interno del range [0kW – valore nominale] e lo si fa propagare.

Per ogni priorità qui esposta vale che nel caso in cui un risultato non sia coerente con i parametri richiesti si registra che il consumo è impostato a 0 kW.

Una volta che è stata determinata la priorità di utilizzo del motore si procede con la seconda parte della logica combinatoria, destinata a verificare il corretto funzionamento dell'impianto. Inizialmente un limitatore – con soglia superiore e inferiore - assicura che errori di calcolo non forniscano risultati impossibili. Quindi ci sono una serie di porte AND e OR.

- Il primo AND è verificato se: la priorità selezionata è la numero 5, i *pressure switch* segnalano un allarme, il motore è in RUN, il pulsante di accensione è impostato su OFF e il motore non è in bypass;

- Il secondo AND è verificato se: il motore è in RUN, è presente almeno un fault, il Bypass è disponibile ma non abilitato.

Questi due AND vengono messi in OR tra di loro e il risultato viene utilizzato come bit di selezione all'ultimo multiplexer, il quale se posto a 1 mostrerà al pannello una potenza assorbita nulla (0 kW), altrimenti quella calcolata ai passi precedenti.

6. Conclusioni e progetti futuri

Alla consegna del lavoro, il cliente è rimasto soddisfatto delle capacità ingegneristiche del team di DSG Automation, e da allora le due aziende hanno avviato una collaborazione che ha portato a nuove opportunità. Grazie alla sua modularità e generalità, il codice sviluppato ha potuto essere adattato a nuove esigenze con facilità come in parte dimostrato in questa tesi.

Il tirocinio mi ha permesso di approfondire temi che non ho potuto considerare durante l'università come i protocolli di comunicazione o i motori elettrici e di applicare le conoscenze fornite dal corso del professor Vitturi - *Laboratorio di automazione industriale* - in modo da comprendere i passi percorsi nello sviluppo di un progetto ampio come questo. Sebbene l'ambiente di sviluppo e i dispositivi utilizzati fossero diversi da quelli del corso, la mia conoscenza preliminare dei concetti chiave ha fatto sì che potessi adattarmi velocemente ad un nuovo contesto. Questa esperienza mi ha inoltre fatto conoscere persone molto preparate che mi hanno aiutato a risolvere i miei dubbi e che mi hanno fatto conoscere alcuni aspetti del mondo dell'automazione industriale. Oltre al progetto riguardante la mia tesi non ho potuto evitare di essere esposto alle dinamiche interne dell'azienda, facendomi apprezzare l'impegno, la capacità, l'umiltà di tutti i collaboratori e le capacità manageriali del titolare.

Per quanto riguarda gli sviluppo futuri, già da ora si sta adattando il codice per implementarlo su una nuova nave più grande; la modifica di alcune delle sue aree ha inevitabilmente portato ad attuare alcune modifiche sia sul codice dei PLC che all'interfaccia grafica del supervisore. Il lavoro che sta svolgendo ora il team di DSG Automation consiste nel trovare le differenze e aggiornare il codice, sulla base dei nuovi diagrammi P&I.

Un'ulteriore possibile modifica, che è ancora in fase di studio, è la creazione di una condizione di *pressione positiva* nei corridoi delle aree delle cabine. Questa necessità è nata in seguito alle conseguenze dell'emergenza COVID19 dove purtroppo proprio un'imbarcazione di questo tipo ha visto sviluppare uno dei primi focolai del virus SARS-COV-2 nel marzo 2020. Con questo meccanismo si creerebbe un ambiente in cui l'aria delle cabine rimarrebbe confinato all'interno delle stanze, rendendo più complicato lo scambio tra cabine adiacenti e con esso la circolazione di virus e batteri.

Bibliografia

S. Vitturi (2022). *Dispensa del corso Laboratorio di Automazione Industriale.* Padova

Sitografia (ultima consultazione: dicembre 2022)

Informazioni generali sui PLC e altri dispositivi Schneider

Modicon M340 Scheda Dati, Schneider Electric
https://www.se.com/ww/en/product/BMXP3420302/processor-module-modicon-m340-automation-platform-max-1024-discrete-+-256-analog-i-o-canopen/

Altivar 61 Programming Manual, Variable speed drives for synchronous motors and asynchronous motors, Schneider Electric
https://www.se.com/ww/resources/sites/SCHNEIDER_ELECTRIC/content/live/FAQS/242000/FA242897/en_US/ATV61%20Programming%20Manual.pdf

Altivar 61 Installation Manual, Variable speed drives for synchronous motors and asynchronous motors, Schneider Electric
https://download.schneider-electric.com/files?p_enDocType=User+guide&p_File_Name=ATV61S_Installation_manual_EN_1760643_06.pdf&p_Doc_Ref=1760643

PowerLogic EGX100 Gateway Ethernet, Caratteristiche del Prodotto, Schneider Electronic
https://www.se.com/it/it/product-range/1432-powerlogic-egx100/#overview

Modbus

Modbus Protocol, Modbus Organization
https://modbus.org/specs.php

Modbus Protocol Specification, Modbus Organization
https://modbus.org/docs/Modbus_Application_Protocol_V1_1b3.pdf

Interfaccia di rete CANopen

CANopen – The standardized embedded network, CAN in Automation (CiA)
https://www.can-cia.org/can-knowledge/canopen/canopen/

Advantys STB Modulo di interfaccia di rete CANOpen, Guida alle Applicazioni, Schneider Electric
https://download.schneider-electric.com/files?p_enDocType=User+guide&p_File_Name=31004621K01000.pdf&p_Doc_Ref=31004621K01000

Ambiente di sviluppo Unity Pro

Start Up Guide for Unity Pro, Schneider Electric
http://users.isr.ist.utl.pt/~jag/aulas/api15/docs/PLC_1_Unity_Startup.pdf

Unity pro operating modes, Schneider Electric
https://media.distributordatasolutions.com/schneider/2016q2/841f1ce646b46730f889d0e1cbe771ef7ef014d7.pdf

Indice Unity Pro XL, Schneider Electric
Interno all'applicazione

Motori Elettrici

Induction motor, G. R. Slemon, Enciclopedia Britannica,
https://www.britannica.com/technology/electric-motor#ref152139

ATS01 Avviatore statico per motore asincrono, dettagli del prodotto, Schneider Electric
https://www.se.com/it/it/product/ATS01N112FT/avviatore-statico-per-motore-asincrono-ats01-12a-110-480v-15-55-kw/

Variatore di Velocità ATV630, scheda dati del prodotto, Schneider Electric
https://www.se.com/it/it/product/download-pdf/ATV630D90N4

Ringraziamenti

Desidero inizialmente ringraziare il professor Stefano Vitturi che ha saputo indirizzarmi verso un'azienda e un tirocinio così interessanti e preziosi per chi come me si sta avvicinando al mondo del lavoro.

Ringrazio tutti i colleghi di DSG Automation che mi hanno seguito in questo percorso, in particolare Gianluca Danesin, Michele Vianello e Alberto Masiero, i quali hanno saputo darmi un supporto fondamentale fornendomi materiale per la stesura della tesi e spiegazioni estensive sui punti più cruciali o di difficile comprensione. In particolare, vorrei ringraziare Gabriele Favorido che, avendo intrapreso un percorso simile al mio nella stessa azienda con pochi mesi di anticipo è stato di fondamentale importanza per fugare dubbi di natura tecnica e personale sul passaggio dal mondo universitario a quello del lavoro.

Ringrazio la mia famiglia che ha atteso a lungo questo momento ma ha saputo supportarmi al meglio. Senza di voi non avrei potuto fare nulla di tutto ciò.

Ringrazio gli amici. Nicola Zennaro, per l'aiuto incredibile fornito durante gli anni dell'università, nello studio e nel prendere le decisioni importanti. Insieme a lui ringrazio Nicola Persona, Giulia Pilon, Marta Sordo, Age, Noemi Calonego, Adriano Canton, Alessandro Lorenzetto per la compagnia di tutti i giorni e le risate.

Infine, ringrazio Giulia, senza la quale questa tesi probabilmente non esisterebbe, per sopportarmi sempre e per avermi fatto capire quanto è importante impegnarsi per raggiungere traguardi come questo.