UNIONE EUROPEA
Fondo Sociale Europeo
Fondo Europeo di Sviluppo Regionale

Ministero dell'Istruzione, dell'Università e della Ricerca

PON RICERCA E INNOVAZIONE 2014 - 2020

Avviso 1735 del 13.07.2017 MIUR

Progetti di Ricerca Industriale e Sviluppo Sperimentale nelle 12 Aree di Specializzazione individuate dal PNR 2015-2020

Scheda tecnica e prestazionale del sistema di acquisizione

Rapporto Tecnico di Ricerca Industriale D4.4

COMESTO
Community Energy Storage
Gestione Aggregata di Sistemi d'Accumulo dell'Energia in Power Cloud

Avviso	Avviso 1735 del 13.07.2017 MIUR
Codice progetto	ARS01_01259
Nome del progetto	Community Energy Storage Gestione Aggregata di Sistemi di Accumulo dell'Energia in Power Cloud
Acronimo	ComESto
Documento	D4.4
Tipologia	Rapporto Tecnico
Obiettivo Realizzativo	OR4
Attività Realizzativa	A4.4
Soggetti Beneficiari Proponenti	UNICAL, SPINTEL
Elaborato (Nome, Cognome – Soggetto Beneficiario)	Anna Pinnarelli, Gaetano Polizzi, Maurizio Vizza - UNICAL Pasquale Cucunato, Floriano De Rango – SPINTEL
Verificato (Nome, Cognome – Soggetto Beneficiario)	Anna Pinnarelli – UNICAL Floriano De Rango – SPINTEL
Approvato (Nome, Cognome – Soggetto Beneficiario)	Membri del PEB

INDICE

Indice delle figure

Indice delle tabelle

Abbreviazioni ed acronimi

Abbreviazione/Acronimo	Testo Esteso
BMS	Battery Management System
CAN	Control Area Network
CRC	Cyclic redundancy check
DAB	Dual Active Bridge
DBS	DC Bus Signaling
HB	Heart Beat
JSON	Java Script Object Notation
NMT	Network Management
PDO	Process Data Object
QoS	Quality of Service
SDO	Service Data Object
SOC	State of Charge

EXECUTIVE SUMMARY

Il presente documento, deliverable del progetto ComESto, sintetizza i risultati delle attività condotte nell'ambito dell'attività 4.4 ("Individuazione di soluzioni architetturali e tecnologiche della nanogrid per la gestione di più sistemi di accumulo, convenzionali e no, integrati con più sistemi di generazione da FR") dell'Obiettivo Realizzativo 4 (OR4).

Obiettivo della attività A4.4 è la definizione, progettazione e realizzazione del prototipo di sistema per acquisizione dati di misure elettriche ed energetiche per la gestione di una nano-grid e sistemi di accumulo convenzionale (batteria a ioni di litio, batteria a flusso, supercapacitori) e non convenzionale (accumulo sotto forma di idrogeno, accumulo sotto forma di biodiesel, accumulo idrico, accumulo termico e Vehicle to Grid). L'attività prevedeva la definizione, la progettazione e la realizzazione di un prototipo capace di effettuare misure elettriche e termiche per la gestione di una nanogrid. Il sistema è da sviluppare considerando diversi fattori sia dal punto di vista hardware sia dal punto di vista software. Particolare attenzione è stata rivolta all'interfacciamento con i sensori di campo che dovranno rilevare le misure elettriche e termiche. Molto importante è stato lo studio e la progettazione delle interfacce di comunicazione che devono essere semplici e raggiungibili anche attraverso interrogazioni remote.

In particolare, l'attività prevedeva:

• Definizione ed implementazione del sistema SdM1 (misure elettriche sistemi di accumulo convenzionali (batterie a ioni di litio, Supercapacitori, Batterie a flusso);

• Definizione ed implementazione del sistema SdM2 (misure elettriche e termiche sistemi di accumulo non convenzionali (Power-to-Hydrogen, accumulo sottoforma di biodiesel, accumulo termico, V2G);

• Definizione ed implementazione dell'Energy box, sistema che acquisisce le informazioni da SdM1 e SdM2 per attuare la logica di controllo e gestione locale secondo i modelli di DR e di ottimizzazione;

• Definizione ed implementazione dell'Energy box, sistema di interfaccia di comunicazione verso l'esterno (verso la piattaforma, il DSO e altro utente per una gestione peer to peer).

Pertanto, l'attività ha visto la realizzazione dei prototipi dei sistemi SdM1 e SdM2 quali sistema di interfaccia per la comunicazione dei dati locali di utenza per la parte dei sistemi di accumulo convenzionale e non convenzionale attraverso il gateway Eugenio, che costituisce l'Energy box di cui ai punti precedenti, oggetto di sviluppo dell'attività A4.1, alla piattaforma ComESto.

Pertanto, è possibile individuare quattro punti principali su cui si è focalizzata l'attività A4.4:

1) Definizione dell'interfaccia di comunicazione
2) Configurazione dei sistemi di acquisizione SdM1 e SdM2;
3) Implementazione dei protocolli di comunicazione SdMX verso il sistema di accumulo e verso il gateway Eugenio e quindi la piattaforma ComESto;

4) Testing dei sistemi SdM1 e SdM2.

1 DEFINIZIONE DELL'INTERFACCIA DI COMUNICAZIONE SDM1 E LE TECNOLOGIE DI ACCUMULO CONVENZIONALE

Il sistema SdM1 è stato sviluppato considerando diversi fattori sia dal punto di vista hardware sia dal punto di vista software. Particolare attenzione è stata rivolta all'interfacciamento con i sensori di campo che dovranno rilevare le misure elettriche e termiche. Molto importante è stato anche lo studio e la progettazione delle interfacce di comunicazione che dovranno essere semplici e raggiungibili anche attraverso interrogazioni remote.

In particolare, l'attività si è articolata in:

- Analisi dello stato dell'arte sui sistemi di acquisizione dati nell'ambito delle microgrid/nanogrid;
- Definizione, progettazione e realizzazione del prototipo del sistema di acquisizione misure elettriche e termiche dei sistemi di accumulo convenzionale, SdM1.

1.1 Stato dell'arte su sistemi di acquisizione dati in nanogrid

L'utilizzo della nanogrid, o più in generale delle risorse rinnovabili in piccola scala, necessita di una struttura di controllo della generazione diversa da quella classica. È per questo motivo che si sta sviluppando sempre di più una struttura di gestione distribuita dell'energia elettrica che, tra le altre cose, necessita di un rapido e continuo scambio di informazioni tra le componenti del sistema. È infatti necessario monitorare le grandezze in gioco per poter gestire in maniera ottimale tutti i nodi della rete, i quali però sono di natura diversa e comunicano quindi in modo diverso. La ricerca di una certa uniformità di comunicazione (standardizzazione) ha portato allo sviluppo di protocolli di comunicazione standardizzati. Negli ultimi anni i protocolli che si sono maggiormente diffusi in ambito industriale sono quello Ethernet, il Wi-Fi e quello fieldbus.

Il protocollo Ethernet è quello più sviluppato ed è usato per mettere in comunicazione un numero non elevato di nodi. Nel corso degli anni sono nate molte varianti ma essenzialmente si tratta di un sistema di tecnologie necessarie per creare una Local Area Network, ovvero LAN.

Il Wi-Fi invece è una tecnologia che si sta evolvendo in modo sempre più rapido negli ultimi anni perché permette, differentemente dall'Ethernet, di bypassare il mezzo fisico del cavo e quindi collegare più facilmente un numero più elevato di nodi anche a distanze considerevoli. Le principali criticità di questa tecnologia sono affidabilità e rapidità, entrambe minori rispetto ad altre possibili soluzioni. L'avanzamento della tecnologia degli ultimi anni si è infatti concentrato sulla risoluzione di queste due criticità ed ha portato ad ottenere risultati eccellenti sotto entrambi i punti di vista.

Con il termine bus di campo (fieldbus) si indicano i protocolli standard di comunicazione seriale tra i diversi dispositivi (nodi) costituenti il sistema. La comunicazione tra i nodi è gestita in modo diverso tra i protocolli in funzione del tipo di bus di campo ed il modello ISO/OSI è il riferimento per la loro classificazione. La struttura dei fieldbus può variare in funzione soprattutto della tipologia di relazione tra i nodi, si possono incontrare bus che seguono il modello Master-Slave, Token Ring o Multimaster.

Per quanto riguarda la prima parte di questa attività ci si è soffermati sull'approfondimento sia delle misure energetiche elettriche effettuate attraverso la nanogrid o attraverso smart-meter, sia sulla comunicazione della nanogrid con i diversi dispositivi che integra e deve coordinare; in particolare ci si è concentrati sullo studio dei protocolli di comunicazione tra nanogrid e sistemi di accumulo convenzionali, batteria a ioni di litio, batteria a flusso e supercapacitore rispettivamente. Lo studio dei protocolli di comunicazione con i sistemi di accumulo è fondamentale per cercare di ottimizzare la comunicazione delle diverse tipologie di sistemi di accumulo che possono coesistere in un nanogrid, magari uniformandoli e facendoli comunicare usando un unico protocollo comune.

Gli smart meter sono un componente fondamentale per il corretto funzionamento di una generica smart grid. Essi servono per monitorare i flussi di energia dei singoli utenti (consumer o producer che siano) in modo tale da fornire tutti i dati necessari al dispacciamento corretto ed anche ottimizzato dell'energia. Un esempio può essere un caso di scompenso energetico zonale che può essere rilevato da uno smart meter il quale comunica tempestivamente (tramite Wi-Fi) direttamente al centro di gestione (ad esempio l'aggregatore o il gestore della comunità energetica) i dati relativi alla situazione energetica. A questo punto attraverso determinati modelli di ottimizzazione si gestisce il trasferimento di energia in modo tale da compensare il deficit energetico rilevato.

Per quanto riguarda i sistemi di accumulo considerati è stato riscontrato tra i prodotti in commercio l'uso di protocolli principalmente di tipo fieldbus, tra i quali spiccano il Modbus ed il CANbus i quali sono protocolli seriali che garantiscono, entrambi, una certa sicurezza di integrità del messaggio ma si differenziano per alcune caratteristiche.

Il Modbus è un protocollo seriale che definisce il formato e la modalità di comunicazione tra un "master" che gestisce il sistema e uno o più "slave". Un limite del Modbus è che si possono connettere un master e fino a 247 slave sul bus. La lunghezza massima del bus è funzione della velocità di trasmissione (bit Rate) ma generalmente non supera i 1200m. Questa caratteristica è derivata direttamente dal tipo di comunicazione seriale scelta ovvero RS-485 ed è in comune anche con il CANbus. Il formato dei messaggi Modbus è 8, N, 1, che vuol dire: 8 bit di dati, nessun controllo di parità e con 1 bit di stop. Il master gestisce la linea e può comunicare con uno slave per volta oppure, in modalità broadcast, con tutti gli slave contemporaneamente. Nell'identificativo del messaggio, quindi, verrà passato per prima cosa un bit di identificazione del nodo slave che andrà da 0 a 247, con 0 che indica il messaggio broadcast mentre i numeri da 1 a 247 sono gli indirizzi degli slave. Si tratta di una logica domanda risposta tra master e slave, il messaggio inviato dal master contiene un codice funzione che identifica quale informazione deve essere inviata dallo slave che riceve.

I limiti di logica di questo protocollo hanno portato alla scelta di utilizzare (se possibile) per tutti gli apparati di storage un unico bus di comunicazione che però utilizza una struttura più flessibile. Il protocollo in questione, anche uno dei più diffusi nel settore industriale, è il CANOpen.

Il CANOpen è un protocollo generato a partire dal CANbus. Il CANbus, Controller Area Network, è nato come protocollo di comunicazione seriale che si integrava con il controllo distribuito e garantiva un certo livello di sicurezza ed inizialmente fu studiato per consentire la comunicazione fra i dispositivi elettronici sugli

autoveicoli. Il CANbus è un protocollo di comunicazione seriale che ha come peculiarità un'elevata capacità di resistere ai disturbi elettromagnetici e mantenere quindi integro il messaggio. Inoltre, è caratterizzato anche dal plug-and-play, ovvero dalla possibilità di aggiungere (o eliminare) nodi senza intaccare il funzionamento della comunicazione. È un bus di tipo lineare e multi-cast, che prevede la connessione di più organi comunicanti (i nodi). Per quanto riguarda le specifiche generiche del CAN (versioni 1.2 e 2.0) esse ricadono nei primi due livelli del modello ISO/OSI ovvero physical layer e data link.

Nel protocollo CANbus tutti i nodi sono in grado di inviare e ricevere messaggi, si tratta quindi di una struttura di tipo Multimaster. I messaggi hanno una determinata struttura e contengono, oltre ai byte di dati, anche dei byte che permettono, tra le altre cose, di gestire le priorità sul bus (identifier), di codificare il messaggio (stuffing) e di effettuare il controllo di eventuali errori (CRC). Per questo motivo il CAN è un protocollo message-based, ogni nodo prima di inviare un eventuale messaggio verifica che il bus sia libero, dopodiché parte il messaggio e, se è presente un conflitto, esso viene risolto attraverso l'identificativo seguendo una regola di priorità.

Il CANbus è definito come un protocollo differenziale, ed è proprio questa sua caratteristica a renderlo meno soggetto a disturbi elettromagnetici. Sul bus si definiscono quindi due linee CANH e CANL che sono quelle su cui viaggia il messaggio sotto forma di differenza di tensione. Tra le due linee vanno inserite due resistenze di terminazione da 120Ω agli estremi del bus che servono per evitare riflessioni di segnale.

Va sottolineato che ogni nodo va messo in comunicazione col bus mediante un transceiver che ha il compito di "tradurre" le differenze di tensione in bit, un bit basso (valore 0) corrisponde ad una differenza di tensione misurata diversa da 0, un bit alto (1) viceversa. Per poter implementare la comunicazione CANbus attraverso l'utilizzo di opportuni microcontrollori, sono stati analizzate le diverse tipologie di transceiver, che sono funzione del bus, in base alle dimensioni di quest'ultimo ed alle tensioni in gioco i transceiver avranno a che fare con livelli differenziali diversi. Per le applicazioni riguardanti il nostro campo di attività è usualmente adottato transceiver a 5V o a 3.3V i quali godono di interoperabilità (Fig. 1).

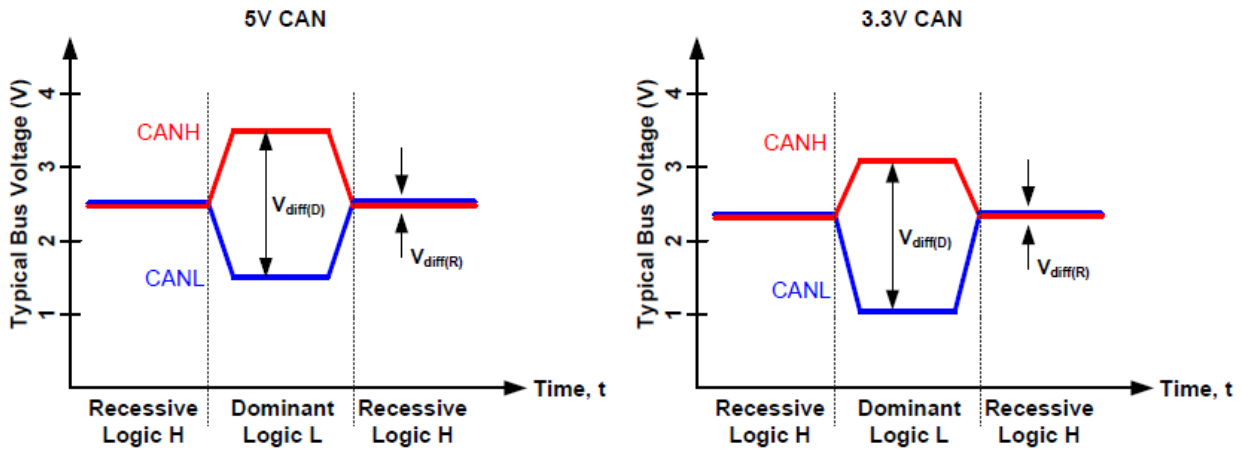


Figura 1 Livelli tipici di tensione transceiver 3.3 V e 5 V CAN bus

Questa caratteristica di interoperabilità è dovuta al fatto che la tensione di modo comune (ovvero i livelli di tensione per CANH e CANL quando il differenziale è nullo) per il transceiver 3.3 V è di circa 2.3 V, mentre

generalmente dovrebbe essere Vcc/2. In questo modo sullo stesso bus possiamo avere tensioni leggermente differenti (0.2 V) che quindi possono essere riconosciute da entrambi i transceiver.

1.2 Il protocollo CANOpen

Partendo dal CANbus e dal suo impiego nel settore dell'automazione, si è tentato di creare uno standard come evoluzione del CANbus che definisse, oltre ai primi due livelli dell'ISO/OSI, anche il settimo, il livello applicativo. Da qui è nato il CANOpen. Il CANOpen si è rivelato essere un protocollo efficiente ed a basso costo, che mantiene inalterate le caratteristiche positive del CANbus andando ad aggiungere il dettaglio della configurazione dei parametri che determinano la comunicazione del nodo, come la relazione di un identificativo con una determinata categoria di messaggi. In ogni caso tutte le configurazioni disponibili sono opzionali così che su un singolo dispositivo si possano implementare solo quelle necessarie. Tra le caratteristiche configurabili nelle librerie del CANOpen ci sono i messaggi che vengono usati per lo scambio di dati e per la gestione della comunicazione. I primi due sono i PDO (Process Data Object) e gli SDO (Service Data Object), mentre per la gestione possono essere usati gli NMT e gli heartbeat.

Dopo aver studiato le caratteristiche e definito il CANbus come protocollo, si è passati ad effettuare qualche test preliminare. Il primo passo effettuato è stato quello di testare il funzionamento del protocollo sui microcontrollori C2000 della Texas Instrument. Per questo passaggio è stato sufficiente utilizzare 2 kit F28379D messi in comunicazione mediante i pin CANH e CANL (e ground) già presenti sulle schede. I transceiver dei due microcontrollori sono da 3.3 V ed hanno già integrata la resistenza di terminazione, il bus è stato quindi completato con dei semplici jumper da una scheda all'altra. Su un microcontrollore è stato inserito un codice per l'invio di messaggi CAN mentre sull'altro un codice per la ricezione. Va sottolineata la necessità di studiare bene le caratteristiche da dare al proprio bus e come queste siano strettamente connesse ai settaggi di inizializzazione di periferica che vengono inseriti nei codici. Questi test sono stati fondamentali per comprendere meglio dal punto di vista pratico il protocollo e, soprattutto, come questo va inizializzato ed usato sul microcontrollore in uso (comandi, clock, GPIO da usare, ecc..).

Il passo successivo è stato quello di testare sul microcontrollore inviante un codice che è stato appositamente scritto per l'invio di un messaggio contenente il valore di un contatore variabile nel tempo. Questo contatore aumenta (o diminuisce) dopo un certo periodo di tempo scelto e quindi verrà inviato al secondo microcontrollore un valore diverso ogni volta che il contatore modifica il suo valore. Sul ricevente invece sarà caricato un codice scritto per la ricezione e la stampa dello stesso messaggio così da verificare la correttezza dei codici. Questi test sono stati utili soprattutto per approfondire alcuni concetti che riguardano l'inizializzazione della periferica CAN sul microcontrollore in questione; va sottolineato che comunque si tratta di messaggi generici e non sono stati utilizzati riferimenti al dizionario oggetti CANOpen ma solo a quelle che sono le caratteristiche del semplice CANbus.

Il prossimo passo è stato quello di mettere in comunicazione il sistema di storage con il microcontrollore per verificare la correttezza della comunicazione anche su altri dispositivi che usano il CAN, inoltre per questo passo

è stata approfondita maggiormente la libreria CANOpen implementata sullo storage. L' obiettivo è di capire come utilizzare al meglio le potenzialità del protocollo e, parallelamente, come integrarlo in modo ottimale nella logica DBS (si veda D4.1) in modo tale da poter leggere ed utilizzare le informazioni che il sistema di accumulo invia senza però inficiare il funzionamento e le tempistiche della logica di controllo.

A tal proposito sono state ipotizzate le grandezze elettriche relative allo storage che sono necessarie per la gestione ottimale del convertitore di interfaccia dello stesso sistema di accumulo e per le verifiche relative allo stato del sistema di accumulo e della comunicazione con lo stesso. La grandezza che sicuramente sarà necessario inviare attraverso un BMS (Battery Management System) è lo stato di carica della batteria (SOC) mediante il quale si potrà gestire il ruolo dello storage all'interno della logica DBS. A tal proposito saranno importanti anche dati relativi ai valori di tensioni e correnti e, se possibile, di setpoint di carica e scarica ottimale dello storage. Quelli appena elencati sono dunque i dati utili per la gestione dell'accumulo: SOC, Tensione, Corrente, Setpoint di carica e di scarica.

Per quanto invece riguarda il monitoraggio della condizione dello storage sarà necessario che il BMS invii delle informazioni come lo stato (ON/OFF, carica/scarica) e soprattutto dei messaggi di Warning/Error/Foults dettagliati che indichino non solo lo stato di allarme ma anche da cosa questo è causato, ad esempio overvoltage, overtemperature, CAN_communication, ecc. Sarà fondamentale, in ultimo, ottenere dal BMS un parametro che renda possibile la verifica dell'integrità della comunicazione, in modo tale da non trovarsi a lavorare con valori errati, un esempio a tal proposito potrebbe essere un Keep Alive ovvero un valore crescente che viene inviato per valutare appunto che il BMS sia 'ancora vivo'.

1.3 Configurazione del sistema di acquisizione per sistema di accumulo convenzionale

Per quanto riguarda il sistema nanogrid, nonostante il CANOpen non richiede un elemento master essendo un protocollo "multi-master", viene comunque individuato come elemento principale del sistema di acquisizione il "modulo base" inteso come nodo che colleziona tutte le informazioni degli altri dispositivi (sistemi di storage) per un'elaborazione sia locale che "remota" trasferendo le informazioni sulla piattaforma ComESto. Come mostrato in Fig. 2, tutti gli elementi vengono quindi connessi sullo stesso bus (CAN_H e CAN_L), realizzato con modalità "twisted", con bit-rate di 125kbps e per tramite del proprio transceiver, isolato per evitare problematiche di interferenze elettromagnetiche usuali in questo tipo di applicazioni.

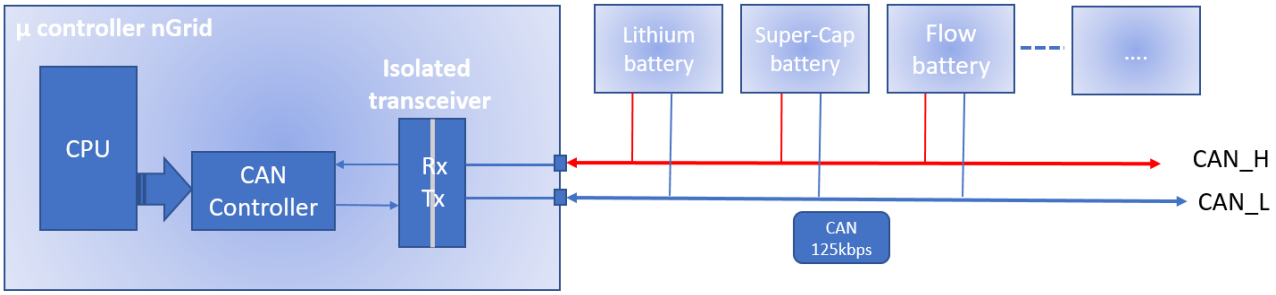


Figura 2 Configurazione del sistema di acquisizione per sistema di accumulo convenzionale

Di seguito vengono elencate le specifiche generali dell'architettura di comunicazione:

- Protocollo CANOpen basato su CAN 2.0A con identifier a 11bit e sample time 85-90%;

- Bitrate di 125 kbps;

- Transceiver ISOLATI a 3.3 V o a 5 V;

- Il nodo master nella rete sarà il microcontrollore del modulo principale della nanogrid, con node-id 0x3f, mentre i diversi sistemi di accumulo saranno numerati progressivamente (ad ese. con BMS basato su CANOpen il BMS ha id 0x01.);

- Come da specifica CANOpen gli NMT devono essere inviati al nodo 0x00 che è il NMT-Master ed il contenuto del messaggio deve contenere il node-id del nodo. Gli NMT saranno utilizzati per settare lo stato Operational/Stopped/Pre-Op ecc;

- Per quanto riguarda il contenuto dei messaggi, negli 8 bytes (max) di un singolo messaggio PDO, al fine di risparmiare banda, potranno essere "accorpate" e trasferite più grandezze acquisite dal sistema (ese. SOC, STATUS, CURRENT, VOLTAGE ecc.);

- I messaggi di tipo node monitoring (node guarding/heartbeat) saranno usati come Heartbeat o come RTR, in funzione della dinamica di comunicazione scelta.

1.3.1 Batteria a ioni di litio

Il sistema di storage a litio è composto da una struttura modulare che presenta un rack a torre con 5 elementi batteria collegati elettricamente in serie. Ogni singolo elemento batteria implementa il proprio sistema di comunicazione internamente mediante apposito hardware che gestisce la fase di acquisizione ed elaborazione delle grandezze "fisiche". I singoli sistemi di comunicazione comunicano con il sistema principale utilizzando uno schema Master-Slave (Fig. 3) che è quello che è connesso al modulo principale della nanogrid e ne gestisce la comunicazione. Anche questo sistema fa uso di un transciever di tipo isolato.

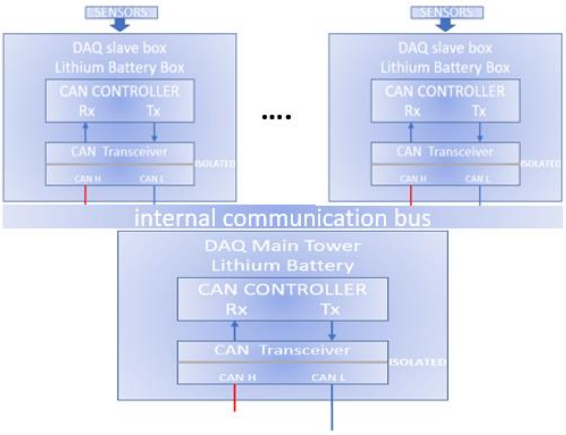


Figura 3 Schema Master-Slave per sistema di accumulo batteria a ioni di litio

1.3.2 Batteria a flusso

La scheda di controllo e supervisione della batteria a flusso integra direttamente l'hardware di comunicazione (isolato) (si veda Fig. 4) e viene quindi direttamente connessa al bus di comunicazione della nanogrid per tramite di apposito connettore denominato J19 le cui connessioni vengono specificate in Tab. 1.

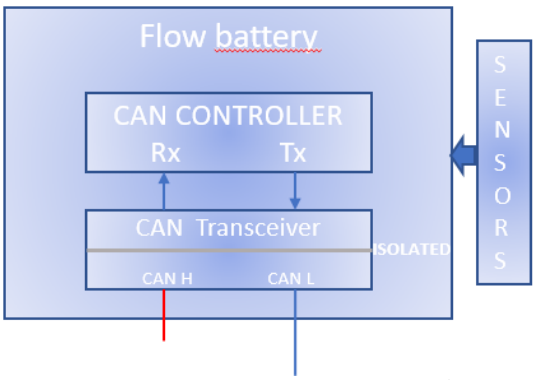


Figura 4 Sistema di comunicazione batteria a flusso

Tabella 1 Specifiche di connessione batteria a flusso

J19 PIN	SEGNALE
1	+5v_can
2	CAN H
3	CAN L
4	Gnd_can

Oltre all'hardware di comunicazione viene anche fornito un sistema di simulazione pensato per verificare e supervisionare esternamente la comunicazione fra il modulo batteria ed il nodo master. In particolare, il sistema è composto da:

• una scheda a microcontrollore di produzione Ala Engineering (Transcriver CanBus isolato, microcontrollore, eeprom etc.) Fig. 5;

• Un software pc (piattaforma Win10) realizzato ad Hoc che comunica con la scheda a microprocessore mediante un adattatore Usb – Rs485;

• il firmware del sistema che realizza le funzioni di comunicazioni CanOpen.

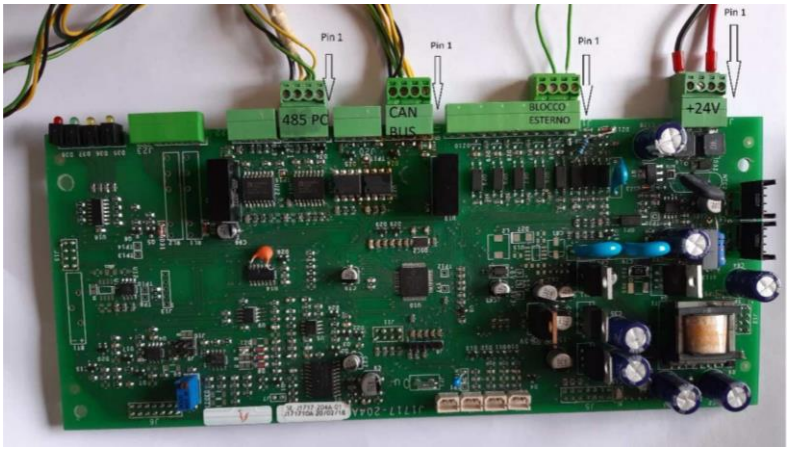


Figura 5 Scheda per sistema di comunicazione batteria a flusso

In Fig. 6 è riportata la schermata per impostare le variabili di configurazione del simulatore.

Per quanto riguarda i parametri di trasmissione "Variabili" possono essere suddivisi in tre gruppi:

1. "Variabili Configurazione Bus". Sono visualizzati in modo tabellare i parametri impostati dal nodo Master. Questi sono i 61 parametri descritti nel documento "Specifiche generali" e sono espressi esattamente come comunicati e presenti nella memoria del microcontrollore per cui dati in formato "signed short". Per le dimensioni le tensioni sono in decimi di V, le correnti di Bus in centesimi di A, i SOC in percentuale, le potenze in W. All' accensione della scheda i parametri vengono caricati dalla eeprom interna e posti nell'area di memoria accessibile via comunicazione CanoOpen, essi possono essere variati runtime (se il nodo non è in modalità di STOP) e verranno memorizzati in eeprom automaticamente dopo cinque secondi dall'ultima variazione. Nel caso uno dei parametri non risultasse coerente la scheda elaborerà il flag di "Errore Parametri" che blocca la macchina a stati e pone nello stato di "Blocco" da cui si esce fornendo il comando di "STOP" per il nodo.
2. "Dal BMS della Batteria". In questo gruppo sono posti i comandi per l'utente per simulare i dati che il BMS della batteria invierà al nostro regolatore. Il sistema di simulazione interno alla scheda elabora questi dati come se provenissero effettivamente dal BMS della batteria ed agisce di conseguenza.
3. "Convertitore Ala": Questo gruppo contiene sia i comandi per l'utente mediante i quali può modificare le variabili di interesse che la visualizzazione dei dati elaborati. I flag di Allarme e Blocco sono già inseriti in gruppi che rappresentano i dati in "unsigned short" presenti nell'area di memoria accessibile mediante i messaggi SDO. I comandi disponibili per l'utente sono quelli che derivano dalle grandezze fisiche e dall'hardware di potenza, nel dettaglio essi sono:

a. Tensione Bus DC: Questo controllo permette di comandare la variabile principale per la determinazione dello stato di lavoro della logica DBS, essa è modificabile liberamente dall'utente per osservare il comportamento del sistema. Oltre che alla gestione della logica DBS, il controllo permette di verificare l'elaborazione dei flag di Allarme e Blocco per le condizioni di SovraTensione e SottoTensione del Bus Dc stesso. Per rendere più precisa la movimentazione e gestire eventuali movimentazioni a gradino è stato introdotto il riquadro "Impostazione V Bus no slide". Mediante il campo numerico presente in esso è possibile impostare un valore in decimi di Volt della tensione di bus simulata che verrà applicato alla posizione della "slide" alla pressione del pulsante "Imposta".
b. Corrente Bus DC: mediante questo controllo è possibile variare il dato di Corrente Erogata (positiva) od Assorbita (negativa) rispetto al bus DC.
c. Tensione e Corrente di Batteria: Questi controlli permettono di modificare i due valori che saranno letti dall'hw di potenza; non influiscono attualmente sul sistema di controllo.
d. Temperature: Questi tre controlli permettono di variare le letture simulate di temperatura che saranno effettuate dall'hw; al variare delle temperature vengono elaborati runtime i flag di Allarme e Blocco associati ad esse. I flag di Blocco influiscono sul comportamento della macchina a stati. Si entra in allarme termico se la temperatura ambiente supera il valore di 55°C per 5 secondi consecutivi e si entra in blocco se supera il valore di 65°C, sempre per 5 secondi. Per La temperatura del trasformatore si ha allarme sopra la soglia di 85°C e blocco sopra la soglia di 105°C, sempre per cinque secondi. Per la temperatura ausiliaria la soglia di allarme è 65°C e quella di blocco 75°C. Tutte le condizioni rientrano se si ha la discesa al di sotto della soglia meno un'isteresi di 5°C per cinque secondi consecutivi.
e. Ingresso Analogico Reference: con questo controllo si può simulare la lettura del dato analogico letto mediante l'ingresso analogico esterno 3Vdc.
f. Condizioni di Allarme: in questo gruppo sono visualizzati i flag di Allarme calcolati runtime dal microcontrollore ed è possibile simulare l'intervento degli allarmi di timeout movimentazione ventole raffrescamento.
g. Condizioni di Blocco: in questo gruppo oltre alle condizioni di blocco calcolate runtime sono presenti tre comandi che permettono all'utente di simulare condizioni che derivano dal'hw di sistema.
h. PDO Riassuntivo: in questo gruppo si hanno a disposizione i dati relativi all'Indice dello Stato di Controllo, al SOC come viene calcolato dal BMS, alle segnalazioni a flag runtime. I dati presenti in questo gruppo vengono inviati automaticamente alla variazione di uno di essi mediante un TPDO del tipo 0x180 + indirizzo nodo, avvisando dunque il master in maniera autonoma di una variazione.

i. Riquadro “Visualizzazione Set Controllo”: In questo riquadro, in funzione dell’attuale stato di controllo, vengono riportati i Set di Controllo ed il valore di Set Potenza del BMS comunicato. Si avranno tre possibilità di visualizzazione: Controllo Tensione Costante (DBS Master), Controllo Corrente Costante (DBS Slave), Controllo Potenza Bus (Centralizzata). Mediante questi dati è possibile verificare le impostazioni di controllo derivanti dai parametri.

j. Nella parte inferiore vi sono dei visualizzatori testuali; si ha:

   i. “Autorizzazione Controllo”: riporta lo stato di controllo impostato mediante l’SDO di indirizzo 6200 sottoindice 1 (dunque “Idle”, “DBS”, “Centralizzata”, “dontcare”)

   ii. “Stato Funzionamento”: Riporta in forma testuale lo stato di controllo attuale come riportato in appendice B.

   iii. “Stato MNT Canopen”: Riporta in forma testuale l’attuale stato del canopen dunque: “INIT”, “Pre Operational”, “Operational”, “Stopped”.

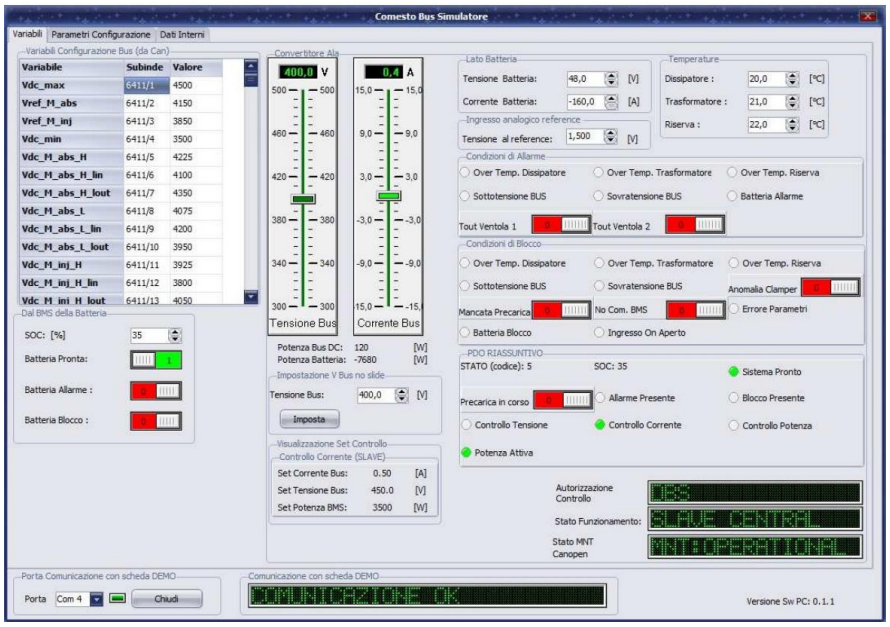


Figura 6 Schermata Simulatore sistema di comunicazione batteria a flusso

In Fig. 7 sono riportati i dati impostabili in fase di configurazione della scheda in eeprom e non presenti nella gestione della comunicazione mediante CanOpen. Attualmente sono presenti quattro gruppi di dati:

1. Id della scheda: mediante questo campo è possibile impostare e variare l’Id della scheda secondo il protocollo CanOpen, il dato è in esadecimale.
2. Tempi di filtraggio: in questo gruppo sono presenti i tempi di filtraggio per il passaggio da uno stato all’altro di controllo al variare di una condizione. I tempi sono in ms. Ad esempio considerando di essere nello stato di “Master Adsorb” una discesa della tensione di bus al di sotto del valore “Vdc_M_abs_L_lout” porta ad uscire dallo stato se la discesa è continuativa per un tempo in ms superiore a “Tempo Uscita Master Adsorb”. Il tempo per il quale si resta nello stato di valutazione del bus è “Tempo

Filtro Valutazione VBUS" e vale per tutte le transizioni. L'ultimo parametro "Tempo invio TPDO Stato" è invece legato all'emissione automatica dell'alive nel TDPO.

3. Set Potenza Per BMS in funzione dello Stato: mediante questo gruppo di parametri è possibile definire il Set di Potenza che la scheda comunica al BMS ad ogni stato funzionale.
4. Autorizzazione Funzionamento: Se lo spunto "Carica in Avvio" è attivo la scheda salva in eeprom il valore del dato "Autorizzazione Controllo" ad ogni modifica e ad ogni riavvio lo reimposta come se fosse appena giunto. In questa maniera se ad esempio si desidera che la scheda parta automaticamente in modalità DBS appena avviata invece che porsi in modalità "Idle" ed attendere l'impostazione del valore 2 nel SDO di indirizzo 6200 sottoindice 1, basta lasciare attivo lo spunto ed impostare almeno una volta il corretto valore. Da questa tab è anche possibile modificare manualmente il dato di Autorizzazione Controllo posto in eeprom.
5. Identificativo: questi sono dati identificativi della scheda e vengono impostati solo dal produttore della scheda.

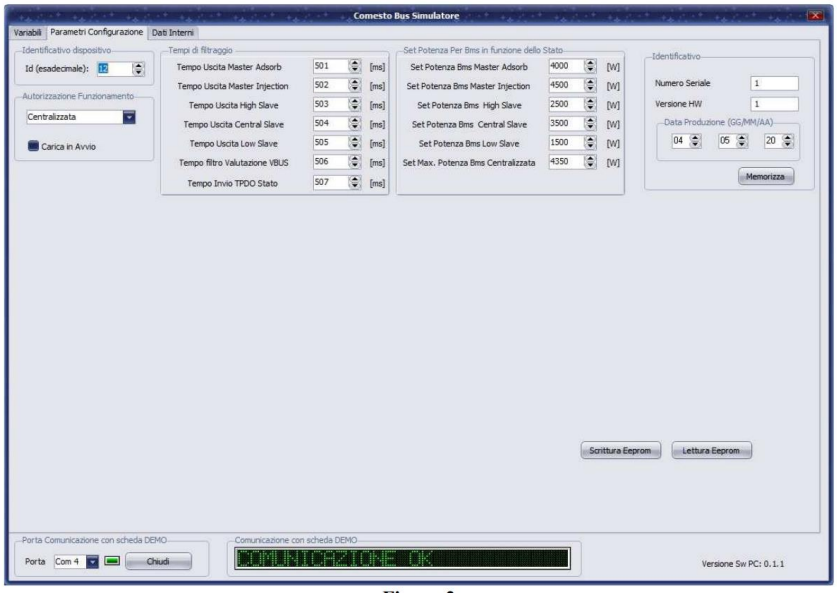


Figura 7 Schermata parametri di configurazione Simulatore di comunicazione

1.3.3 Supercapacitore

Per quanto riguarda il sistema a Supercapacitore (Fig. 8) si è realizzato un'apposita scheda per la supervisione del sistema e la comunicazione che è separata dalla scheda di controllo dell'interfaccia di potenza.

Questa scheda sarà composta dal proprio microcontrollore, tranceiver CAN isolato 5V, seriale ed alcuni ingressi analogici. Il firmware implementerà la macchina a stati definita dalla specifica CANOpen e comunicherà via seriale con la scheda di controllo dell'interfaccia di potenza utilizzando il protocollo MODBUS RTU.

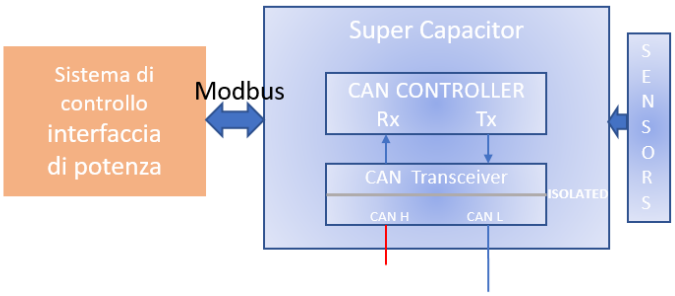


Figura 8 Schema del sistema di comunicazione per Supercapacitore

Sono state implementate delle primitive generali che consentono di leggere e scrivere i parametri in MODBUS RTU ed uno strato di funzionalità che, utilizzando le librerie primitive implementi i seguenti comandi:

- preimpostare l'interfaccia di potenza in modalità MASTER/SLAVE (decentralizzata a logica DBS) usando le soglie definite e le isteresi impostate

- preimpostare l'interfaccia di potenza in modalità CENTRALIZED (controllo in potenza) usando il riferimento analogico fornito dal gestore della nanogrid che verrà trasmesso al DAB via Modbus

- preimpostare l'interfaccia di potenza in modalità IDLE. In tale modalità si è in standby, coincide con il pre-operazionale ovvero switched on disabled, in caso di DAB impostato in modalità IDLE l'eventuale arrivo del NMT OPER non sortirà nessun effetto in quanto, per vincolo di progetto, durante l'impostazione dei parametri si è già in standby mentre non è ammissibile variare la modalità mentre si è in operazione.

- preimpostare l'interfaccia di potenza in modalità dontcare. In tale modalità il DAB insegue un riferimento di corrente fornito via modbus.

- portare l'interfaccia di potenza in stato operazionale. In tale modalità il convertitore inizia a funzionare secondo la modalità preimpostata, a meno che questa non sia impostata a IDLE, nel qual caso ignora il comando

- portare l'interfaccia di potenza in stato non operazionale, il convertitore viene arrestato e deve transitare in modalità switched on disabled, a meno che non lo sia già.

-Derating: è prevedibile una situazione di sovraccarico dell'interfaccia di potenza o dei SC (in qualunque modalità operativa) per cui il converter effettua un derating; può quindi effettuare un derating automatico, ad esempio sta erogando/assorbendo P0 e si verifica un sovraccarico per cui il converter modifica in autonomia la potenza (Pnew=P0 +/- Delta) in automatico e segnala sia la situazione di derating e sia l'entità (Delta), che può essere espresso anche in %.

-Raggiungimento/Incompatibilità di SOC: durante il funzionamento, in qualunque modalità, può accadere che il limite di SOC venga raggiunto, in questo caso viene segnalato il raggiungimento azzerando l'azione di controllo ("derating 100%") e può essere gestito come nel caso precedente.

Nel caso in cui tra il passaggio di modalità ci si ritrovi un SOC incompatibile, ad esempio SOCmin=50 e SOC attuale=40 allora se l'azione di controllo va nello stesso verso dell'incompatibilità allora ricadiamo nel caso precedente. Viceversa, se l'azione di controllo va nel segno opposto dell'incompatibilità allora questa è consentita (il vincolo non si attiva).

La gestione del fault avverrà secondo le seguenti specifiche:

"FAULT DAB"

- fault critico per cui l'interfaccia di potenza non è in grado di funzionare per anomalie o avarie (da elencare tutti i fault critici di questo tipo con descrizione, ragioni e creare dei codici appositi di errore DAB). In tal caso il DAB passerà in modalità pre-operazionale (idle). Tale di insorgenza o possibili cause, possibili contromisure situazione verrà notificata alla nanogrid mediante apposito messaggio EMCY ed esplicitata nello status word di sistema

- fault non critico per cui l'interfaccia di potenza è in grado di funzionare ma a potenza ridotta (es. rilevato trend di surriscaldamento o superate soglie SOC) In tal caso l'interfaccia di potenza andrà in derating. Tale situazione verrà notificata alla nanogrid esplicitandola nella status word di sistema (bit derating on).

Quando il problema dovesse rientrare l'interfaccia di potenza tornerà autonomamente nella modalità che aveva prima, ovviamente segnalandolo nella status word alla sua interfaccia che provvederebbe a notificarlo al gestore della nanogrid.

"FAULT SCHEDA INTERFACCIA"

- fault comunicazione est. (es. non si riceve più i RTR) il dab viene impostato in modalità: (TBD IDLE);

- fault di comunicazione int. (es non si comunica più via MODBUS) è assimilabile ad un fault critico in quanto non si conosce lo stato del dab, si deve inviare EMCY e tornare in stato pre-operazionale. Si noti che lato DAB questo dovrebbe essere riconosciuto come errore critico e portare allo stato pre-operazionale in ogni caso, sia quando non riceve più comandi modbus r/w dalla scheda di interfaccia entro un timeout, sia quando non riesce ad inviare a destinazione il Modbus RESP, infatti in quest'ultimo caso, dopo un timeout la scheda di interfaccia rileva la mancanza di risposte e va in fault smettendo di inviare i MODBUS REQ cosicchè si ricade nel primo caso.

- fault acquisizione analogica segnale di riferimento centralizzato da nanogrid (solo modalità centralizzata) invio EMCY e impostazione dell'interfaccia di potenza in modalità: (TBD IDLE).

- fault di overvoltage dei supercapacitori: l'interfaccia di potenza effettua derating per consentire solo la scarica

- fault di overtemperature dei supercapacitori: assimilabile ad un fault non critico dell'interfaccia di potenza si va in modalità derating per ridurre le potenze.

NOTA sulle reazioni ai fault: queste due ultime situazioni di fault (non critiche) devono essere acquisite dall'interfaccia di potenza mediante la lettura del parametro inviato via modbus STATUSWORD SYS nei bit 11 e 12, infatti tali situazioni sono rilevate via ingressi analogici dalla scheda interfaccia ma l'azione di derating spetta all'interfaccia di potenza.

Nei casi in cui il fault richiede lo spegnimento della potenza (passaggio in IDLE), ad es. nel caso di perdita della comunicazione CAN, la scheda d'interfaccia imposta il parametro modalità a IDLE.

Nel caso di perdita della comunicazione modbus deve essere ovviamente l'interfaccia di potenza a rilevarla ed a transitare in modalità IDLE come già specificato.

Di seguito si elencano le specifiche relative alla comunicazione CAN tra il nodo di accumulo ed il microcontrollore di gestione della nanogrid:

- Il bitrate in uso è di 125 kbps
- Transceiver sia a 3.3 V che a 5 V (isolati).
- Il nodo master nella rete sarà il microcontrollore che gestirà la nanoGrid, con node-id 0x3f, gli accumuli saranno numerati progressivamente con la possibilità che sia necessario più di un node-id per accumulo (nel caso attuale con BMS basato su CANOpen il BMS ha id 0x01 ma gli NMT devono essere inviati al nodo 0x00 che è il NMT-Master)
- Per quanto riguarda il contenuto dei messaggi, negli 8 bytes di un singolo messaggio RxPDO possono essere inserite diverse grandezze per risparmiare banda.
- il riferimento analogico della specifica relativa al modo di controllo centralizzato non potrà essere inviato in formato digitale via CANOpen.

Le grandezze monitorate sono di seguito elencate:

* SOC – State Of Charge (in Wh, ad esempio, oppure in J);

* STATUS (un codice che è 0 se tutto ok o diverso da 0 se c'è un errore, nel qual caso il valore sarà un codice dell'errore). Possibili codici di fault (non esaustivo) sono overtemperature, overvoltage, hardware fault. Saranno indicativaente suddivisi tra critici e non critici. I primi indicano fault che mettono in protezione e distaccano il nodo supercondensatori, i secondi indicano che il nodo continua a funzionare ma applica delle specifiche di potenza rilassate rispetto a quanto richiesto dalla rete in quanto persistono situazioni problematiche (es temperatura alta);

* SOB – State Of Balancing è una specie di SOH ma non è riferito alla "salute" della batteria ma allo stato di bilanciamento tra le varie celle SC in serie;

* % DERATING che indica in che % della potenza o corrente il convertitore sta lavorando a seguito di problemi non critici (es. trend temperatura in salita), rispetto al setpoint desiderato dal gestore;

* TEMP la temperatura del convertitore o dei condensatori a queste, nel sistema gestore, vanno aggiunte tutte le altre grandezze caratteristiche degli altri tipi di accumuli;

Si può definire la specifica che ciascun tipo di accumulo comunichi solamente le grandezze di pertinenza, ad es. i SC invieranno il SOB mentre il litio il SOH. A ciascuna grandezza assegneremo un mapping nei PDO su CANOpen;

-gestione dei soli SDO expedited upload transfer che consentono ad un master di modificare un parametro read/write int (a 8, 16 0 32) su un nodo slave (REQ). Lo slave risponde con un messaggio RES che conferma se l'impostazione del parametro è avvenuta con successo o meno. Data la presenza di campionamenti a 12 bit si assume che i parametri relativi a soglie e delta saranno scambiati come int16;

- sono previsti 34 SDO con codici unici per impostare le 6 soglie di tensione, i 10 riferimenti di corrente, i 12 riferimenti di SOC, i due livelli di isteresi (inf e sup), i due livelli di potenza max e min in modalità centralizzata e la modalità di funzionamento (MASTER/SLAVE/IDLE/DONTCARE);

-per ogni parametro, sia read only che read/write è previsto l'invio allo slave di un SDO di lettura che restituisce al CAN master il valore attualmente impostato per il parametro;

-le operazioni di ricezione dei SDO di impostazione sono ammissibili solamente mentre lo slave è non operazionale in quanto la modifica di una singola soglia a caldo potrebbe provocare anomalie ed interferenze. I SDO di lettura, al contrario, sono sempre disponibili;

- il node-id di uno slave è predefinito ed assegnato, non è previsto riconfigurare il node id via SDO;

- i PDO slave (fino a 8 TPDO, quindi fino a 32 parametri int16) sono predefiniti, non è previsto modificare il mapping dei parametri via SDO;

- non sono previsti PDO master (RPDO);

- è previsto l'invio da master di un messaggio NMT OPER rivolto ad un nodo od all'intera rete (Broadcast) per imporre l'avvio (cod.funzione 0x1) o la cessazione (cod.funzione 0x80) delle operazioni di potenza del nodo(i) coerentemente con le soglie e le modalità preconfigurate o modificate mediante SDO in modalità pre-operazionale. Per evitare situazioni anomale e potenzialmente dannose le soglie e le modalità preimpostabili prima di andare in operazione dovranno avere dei default che potranno essere modificati solamente se le nuove impostazioni saranno ammissibili (vedi condizioni e vincoli di impostazione riportati nel paragrafo descrittivo dei SDO). Qualora si tentasse di iniziare le operazioni di potenza con parametri erronei o non coerenti con i vincoli verranno ripristinati i default o i valori precedentemente impostati prima di andare in operazione.

ATTENZIONE, per evitare errori su valori ammissibili soglie prima che siano tutte impostate è prevista una procedura lato gestore per impostarle in un ordine predefinito.

- è previsto un messaggio RTR inviato da master ad un nodo o in broadcast a tutta la rete. Se un nodo riceve tale messaggio dovrà inviare immediatamente i propri TPDO corredati di node id e opportunamente valorizzati con i più recenti dati di funzionamento definiti dai parametri mappati e concordati. RTR è anche utilizzato come heartbeat bidirezionale, cioè gli slave operativi possono dedurre che il gestore ha cessato di funzionare se non ricevono RTR dopo un certo timeout (da definire) ed in tal caso uscire dalla modalità operazionale fino a ripristino del RTR. A sua volta il gestore può dedurre che un nodo ha smesso di funzionare se a seguito di invio RTR non riceve PDO di risposta dal nodo dopo un determinato timeout.

- per modificare le condizioni operative di uno slave (es. reset o stop) si utilizza sempre il messaggio NMT OPER con il codice funzione opportuno quindi lo slave entra in modalità IDLE ed accetta SDO. Da verificare anche dal punto di vista elettrico (soft start/stop)

1.4 Implementazione del protocollo per la comunicazione con i sistemi accumulo convenzionale

1.4.1 Batteria a ioni di litio

L'implementazione del protocollo di comunicazione tra il microcontrollore che gestisce la nanogrid ed il BMS del sistema di accumulo al litio è partita dall'analisi delle caratteristiche del protocollo implementato sull'accumulo. Una volta identificate tali caratteristiche e analizzata la mappatura dei messaggi è stato modificato il codice presente sul microcontrollore in modo tale da poter interagire con il BMS e quindi leggere o inviare messaggi. Il bus di comunicazione è stato inizializzato impostando un bitrate di 125 kbps ed un sample time del 85-90%. Questi parametri sono stati mantenuti identici nel processo di definizione della comunicazione degli altri sistemi di accumulo.

Il BMS dell'accumulo a litio utilizza per la comunicazione PDO, HB e NMT. Rispetto a quelli che sono gli strumenti generalmente usati nel CANOpen, quindi, non sono usati gli SDO.

I messaggi NMT (Network management) servono per cambiare lo stato del sistema, inviando un NMT al BMS è possibile variare lo stato tra preoperational, stopped o operational. Il format per un NMT, in Fig. 9, prevede di inviare 2 byte di dati contenenti sia lo stato che deve assumere il sistema sia l'indirizzo dello stesso.

Figura 9 Struttura standard del messaggio NMT

Più nel dettaglio, il messaggio è composto da:

- CAN_ID: 0x00, riservato per i messaggi NMT;
- LEN: indica la lunghezza del messaggio, che per gli NMT è 2 byte;
- B0: NMT-Command, Operational/Pre-operational/Stopped;
- B1: Node-ID, l'identificativo del nodo con cui si vuole dialogare sul bus.

Per quanto concerne il sistema di accumulo, la ricezione di un NMT che varia lo stato in operational coincide

con la chiusura del contattore per la gestione della potenza, viceversa una variazione da operational causa la riapertura del contattore e quindi la disconnessione del lato potenza del sistema.

Gli HB (Heartbeat) invece sono dei messaggi usati per monitorare lo stato del sistema di accumulo e sono inviati dal BMS con frequenza elevata per comunicare esclusivamente se si trova in operational/stopped/preoperational. Per quanto concerne l'applicazione studiata è stato scelto di implementare una funzione che invia l'NMT Operational al sistema non appena viene avviato il microcontrollore e che si possa scegliere, in seguito, quando inviare gli NMT di operational e preoperational.

I PDO sono messaggi inviati dal BMS del sistema di accumulo e contenenti i dati come tensioni, correnti, SOC, ecc., nel caso testato il BMS invia una serie di 7 PDO solo quando è in condizioni di operational. Il formato standard dei PDO è rappresentato in Fig. 10.

Figura 10 Struttura standard del messaggio PDO

Un generico PDO è formato da:

- CAN-ID: ID del messaggio, deve essere mappato precedentemente, solitamente è standard (0x480+Node-id, 0x380+Node-id, 0x580+Node-id, etc.);
- RTR: remote transmission request, utilizzabile solamente con i messaggi configurati per accettare richiesta di invio dal master;
- LEN: indica la lunghezza del messaggio, che può arrivare fino al massimo di byte 8;
- DATA BYTES: byte contenenti i dati da inviare.

Il microcontrollore è stato quindi programmato in funzione della mappatura dei PDO (Fig. 11). È stato necessario impostare indirizzi ed indici per PDO e NMT ed inoltre implementare una funzione per la "decodifica" dei dati ricevuti e la stampa via seriale degli stessi così da visualizzare i valori letti. La decodifica dei PDO è necessaria in quanto un singolo PDO contiene più dati, come si evince dalla figura seguente che rappresentano il contenuto dei PDO inviati dal sistema di accumulo.

CAN-ID	DLC	B0-B1	B2-B3	B4	B5	B6	B7
0x380+Node-ID	8	Battery Current	Battery Voltage	SOC	SOH	Battery State	Keep Alive

CAN-ID	DLC	B0-B1	B2-B3	B4	B5
0x400+Node-ID	6	Remaining Capacity	Remaining Energy	Charge State	-

CAN-ID	DLC	B0-B1	B2-B3	B4-B5	B6-B7
0x280+Node-ID	8	Battery Current	Highest Cell Voltage	Lowest Cell Voltage	Average Cell Voltage

CAN-ID	DLC	B0	B1	B2	B3	B4	B5
0x480+Node-ID	6	Lowest Cell Temp	Highest Cell Temp	Average Cell Temp	Average Board Temp	-	-

CAN-ID	DLC	B0	B1	B2
0x500+Node-ID	3	Lowest Cell Resistance	Highest Cell Resistance	Average Cell Resistance

CAN-ID	DLC	B0-B1	B2-B3	B4-B5	B6-B7
0x180+Node-ID	8	Warning	Error	Faults	-

CAN-ID	DLC	B0-B1	B2-B3	B4-B5	B6-B7
0x200+Node-ID	8	Charge Setpoint	Discharge Setpoint	P/N Contactor	Fault mask 1/2/3

Figura 11 Mappatura PDO Litio

È stata valutata la possibilità di legare la lettura dei PDO ad un interrupt legato appunto a questa tipologia di messaggi ma, per comodità, si è scelto di effettuare la lettura dei parametri in modo asincrono e con una frequenza maggiore rispetto a quella di invio. La lettura dei messaggi PDO viene effettuata in un interrupt con periodo di un millisecondo, subito prima dell'esecuzione delle funzioni relative alla logica di gestione del sistema che dipendono proprio dai parametri letti tramite comunicazione CAN. La scelta dei parametri da leggere ed il loro utilizzo all'interno della logica di gestione sono risultato del OR2.

1.4.2 Batteria a flusso

L'utilizzo del CANOpen è stato scelto, come anticipato, anche per le altre tipologie di accumulo. Per quanto concerne le batterie a flusso, in confronto al sistema agli ioni di litio, è stato necessario implementare molte più funzionalità in quanto il bus di comunicazione rappresenta l'interfaccia non solo con il sistema ma anche con il convertitore ad esso relativo. Una delle differenze implementative di maggior interesse, rispetto all'accumulo al litio, è l'utilizzo dello strumento Service Data Object (SDO) in precedenza non necessario. La struttura di un generico SDO è riportata in Fig. 12.

CAN Header	RTR	LEN	Byte 0 COMMAND	Byte 1-2 INDEX	Byte 3 SUB-INDEX	Byte 4-7 DATA

Figura 12 Struttura standard del messaggio SDO

La struttura del messaggio SDO ricorda molto quella del PDO ma ha delle sostanziali differenze. L'SDO è formato da:

- CAN Header: Generalmente indicato anche come COB-ID è determinato dalla funzione che l'SDO svolge (riceve o trasmette) e dall'id del nodo. Il valore è 0x600+"node-id" per SDO in ricezione e 0x580+"node-id" per SDO in trasmissione. Va sottolineato che la definizione di direzione TX/RX è data dal punto di vista del dispositivo. Quindi, per interrogare un dispositivo sulla rete, viene inviato un 0x600+"node-id" e si ottiene un 0x580+"node-id"
- RTR: in questo caso è sempre 0
- LEN: indica la lunghezza del messaggio e per gli SDO è sempre 8
- COMMAND: occupa il primo byte del messaggio e dipende dall'attività che si vuole svolgere (lettura/scrittura) e dalla lunghezza dei dati da scrivere/leggere.
- INDEX: occupa il secondo e terzo byte e rappresenta l'indirizzo in cui è mappato il parametro su cui si vuole agire
- SUB-INDEX: occupa il quarto byte e rappresenta il sotto-indirizzo in cui è mappato il parametro su cui si vuole agire
- DATA: sono i 4 bytes finali all'interno dei quali è contenuto il messaggio che si vuole scrivere. Nel caso si volesse leggere un messaggio questo campo deve essere posto a 0.

Mediante questa tipologia di messaggio è possibile, solamente quando il sistema non è in fase operazionale, la

scrittura dei parametri caratteristici della macchina a stati che governa il funzionamento del convertitore. In funzione della priorità che si vuole dare al sistema di accumulo all'interno della logica DBS vengono modificati i livelli di tensione, corrente e SOC min/max mediante l'utilizzo degli SDO. Anche la lettura di questi parametri avviene utilizzando gli SDO, ed è possibile effettuarla anche durante la fase operazionale. Di fondamentale importanza è stata la mappatura di questi parametri che è riportata in Tab. 2-3.

Tabella 2 Dati in Sola Lettura batterie a flusso

Indice	**Sottoindice**	**Dimensione / Tipo**	**Nome**	**Nota**
6000	01	8 bit / unsigned char	SOC	Valore del dato di SOC in precentuale come comunicato al regolatore dal BMS della batteria
6000	02	8 bit / unsigned char	Flag Stato Runtime	<table><tr><td>Bit 0</td><td>Sistema Pronto</td></tr><tr><td>Bit 1</td><td>Precarica On</td></tr><tr><td>Bit 2</td><td>Allarme Presente</td></tr><tr><td>Bit 3</td><td>Blocco Presente</td></tr><tr><td>Bit 4</td><td>Erogazione in Controllo di Tensione Vbus</td></tr><tr><td>Bit 5</td><td>Erogazione in Controllo di Corrente Ibus</td></tr><tr><td>Bit 6</td><td></td></tr><tr><td>Bit 7</td><td></td></tr></table>
6000	03	8 bit	Stato di Controllo	Valore dello stato della macchina a stati di controllo, i valori saranno definiti durante lo sviluppo della stessa. (es 0 = Idle, 1 = Master Adsorb ,0xFF Blocco etc…)
6401	01	16 bit / signed short	Tensione Bus DC (V*10)	Tensione letta sul bus DC in decimi di Volt (es 4123 = 412,3Vdc)
6401	02	16 bit / signed short	Corrente Bus DC (A*100)	Corrente letta sul bus DC in centesimi di A (es 1023 = 10,23 A), se positivo si è in erogazione verso il bus DC se negativo in assorbimento (si carica la batteria)
6401	03	16 bit / signed short	Tensione di Batteria (V*10)	Tensione di Batteria in decimi di Volt (es 485 = 48,5Vdc)

<table>
<tr><td>6401</td><td>04</td><td>16 bit / signed short</td><td>Corrente di Batteria (A*10)</td><td>Corrente di Batteria in decimi di A (es 923 = 92,3 A), se positivo si sta caricando la batteria, se negativo si sta erogando verso il bus.</td></tr>
<tr><td>6401</td><td>05</td><td>16 bit / signed short</td><td>Tensione Riferimento esterno in mV</td><td>Tensione letta sull’ ingresso analogico di riferimento esterno a fondoscala 3V in mV (es. 1500 = 1,5Vdc)</td></tr>
<tr><td>6401</td><td>06</td><td>16 bit / signed short</td><td>Temperatura Dissipatore in decimi di °C</td><td>Temperatura letta mediante la Ntc posta sul dissipatore in decimi di °C (es 256 = 25,6°C)</td></tr>
<tr><td>6401</td><td>07</td><td>16 bit / signed short</td><td>Temperatura Trasformatore in decimi di °C</td><td>Temperatura letta mediante la Ntc posta sul trasformatore alta frequenza in decimi di °C (es 243 = 24,3°C)</td></tr>
<tr><td>6401</td><td>08</td><td>16 bit / signed short</td><td>Temperatura Riserva in decimi di °C</td><td>Temperatura letta mediante la Ntc di riserva (prevista ma non ancora allocata) in decimi di °C (es 243 = 24,3°C)</td></tr>
<tr><td>6401</td><td>09</td><td>16 bit / unsigned short</td><td>Bitmap degli Allarmi presenti</td><td>
<table>
<tr><td>Bit 0</td><td>Sovratemperatura Dissipatore</td></tr>
<tr><td>Bit 1</td><td>Sovratemperatura Trasformatore</td></tr>
<tr><td>Bit 2</td><td>Sovratemperatura sonda Riserva</td></tr>
<tr><td>Bit 3</td><td>Sottotensione Bus DC</td></tr>
<tr><td>Bit 4</td><td>Sovratensione Bus DC</td></tr>
<tr><td>Bit 5</td><td>Batteria comunica Allarme</td></tr>
<tr><td>Bit 6 - 15</td><td>Non allocati</td></tr>
<tr><td></td><td></td></tr>
</table>
</td></tr>
</table>

6401	10	16 bit / unsigned short	Bitmap dei Blocchi Presenti	Bit 0	Sovratemperatura Dissipatore
				Bit 1	Sovratemperatura Trasformatore
				Bit 2	Sovratemperatura sonda Riserva
				Bit 3	Sottotensione Bus DC
				Bit 4	Sovratensione Bus DC
				Bit 5	Errore nei Parametri (dati sui 6411)
				Bit 6	Batteria Comunica Blocco
				Bit 7	Anomalia Circuito Clamper
				Bit 8	Blocco per Mancata Precarica
				Bit 9	Mancata Comnicazione con il BMS di batteria
				Bit 10 - 15	Non allocati
6401	11	16 bit /signed short	Potenza Istantanea Verso Bus	Valore della potenza istantanea verso bus DC in W.	
6401	12	16 bit /signed short	Potenza Istantanea Verso Batteria	Valore della potenza istantanea verso bus Batteria in W.	
6401	13	16 bit / unsigned short	Set Tensione Bus	Valore dell' attuale set di Tensione di Bus per il controllo in V*10;	
6401	14	16 bit /signed short	Set Corrente Bus	Valore sell' attuale set di Corrente di Bus per il controllo in A*100	
6401	15	16 bit /signed short	Set Potenza Comunicato al BMS	Valore dell' attuale set di Potenza di Batteria per il BMS di batteria in W	

Tabella 3 Dati in Sola scrittura batterie a flusso

Indice	Sottoindice	Dimensione / Tipo	Nome	Valore Default	Nota

6200	1	8 bit	Autorizzazione Controllo	**0x00**	**Valore** / **Funzione** 0x00 – Sistema Vincolato allo stato di controllo Idle, anche dopo aver ricevuto il comando di start nodo si resta nello stato di Idle 0x01 – Gestione DBS: al ricevimento del comando di Start Nodo, il sistema si porta in gestione a logica decentralizzata 0x02 – Gestione Centralizzata: al ricevimento del comando di Start Nodo, il sistema si porta in gestione a logica centralizzata con riferimento analogico. 0x03 – DONTCARE
6411	01	16 bit	Vdc_max	450.0	In decimi di Volt
6411	02	16 bit	Vref_M_abs	415.0	In decimi di Volt
6411	03	16 bit	Vref_M_inj	385.0	In decimi di Volt
6411	04	16 bit	Vdc_min	350.0	In decimi di Volt
6411	05	16 bit	Vdc_M_abs_H	422.5	In decimi di Volt
6411	06	16 bit	Vdc_M_abs_H_lin	4100	In decimi di Volt
6411	07	16 bit	Vdc_M_abs_H_lout	4400	In decimi di Volt
6411	08	16 bit	Vdc_M_abs_L	4075	In decimi di Volt
6411	09	16 bit	Vdc_M_abs_L_lin	4200	In decimi di Volt

6411	10	16 bit	Vdc_M_abs_L_lout	3950	In decimi di Volt
6411	11	16 bit	Vdc_M_inj_H	3925	In decimi di Volt
6411	12	16 bit	Vdc_M_inj_H_lin	4050	In decimi di Volt
6411	13	16 bit	Vdc_M_inj_H_lout	3800	In decimi di Volt
6411	14	16 bit	Vdc_M_inj_L	3755	In decimi di Volt
6411	15	16 bit	Vdc_M_inj_L_lin	3900	In decimi di Volt
6411	16	16 bit	Vdc_M_inj_L_lout	3650	In decimi di Volt
6411	17	16 bit	Vdc_SH_H	4450	In decimi di Volt
6411	18	16 bit	Vdc_SH_H_lin	4400	In decimi di Volt
6411	19	16 bit	Vdc_SH_H_lout	4500	In decimi di Volt
6411	20	16 bit	Vdc_SH_L	4225	In decimi di Volt
6411	21	16 bit	Vdc_SH_L_lin	4400	In decimi di Volt
6411	22	16 bit	Vdc_SH_L_lout	4100	In decimi di Volt
6411	23	16 bit	Vdc_SC_H	4075	In decimi di Volt
6411	24	16 bit	Vdc_SC_H_lin	4075	In decimi di Volt
6411	25	16 bit	Vdc_SC_H_lout	4075	In decimi di Volt
6411	26	16 bit	Vdc_SC_L	3925	In decimi di Volt
6411	27	16 bit	Vdc_SC_L_lin	3925	In decimi di Volt
6411	28	16 bit	Vdc_SC_L_lout	3925	In decimi di Volt
6411	29	16 bit	Vdc_SL_H	3775	In decimi di Volt
6411	30	16 bit	Vdc_SL_H_lin	3650	In decimi di Volt
6411	31	16 bit	Vdc_SL_H_lout	3900	In decimi di Volt
6411	32	16 bit	Vdc_SL_L	3500	In decimi di Volt
6411	33	16 bit	Vdc_SL_L_lin	3500	In decimi di Volt
6411	34	16 bit	Vdc_SL_L_lou	3500	In decimi di Volt
6411	35	16 bit	Imax_HS	-2 50	In centesimi di Ampere
6411	36	16 bit	Iref_HS	-250	In centesimi di Ampere
6411	37	16 bit	Imin_HS	+50	In centesimi di Ampere
6411	38	16 bit	Imax_M_abs	-250	In centesimi di Ampere
6411	39	16 bit	Imin_M_abs	+50	In centesimi di Ampere
6411	40	16 bit	Imax_CS	+250	In centesimi di Ampere

6411	41	16 bit	Iref_CS	+50	In centesimi di Ampere
6411	42	16 bit	Imin_CS	-250	In centesimi di Ampere
6411	43	16 bit	Imax_M_inj	+250	In centesimi di Ampere
6411	44	16 bit	Imin_M_inj	-50	In centesimi di Ampere
6411	45	16 bit	Imax_LS	+250	In centesimi di Ampere
6411	46	16 bit	Iref_LS	+250	In centesimi di Ampere
6411	47	16 bit	Imin_LS	-50	In centesimi di Ampere
6411	48	16 bit	SOCmax_HS	95	In percentuale di SOC
6411	49	16 bit	SOCmin_HS	10	In percentuale di SOC
6411	50	16 bit	SOCmax_M_abs	95	In percentuale di SOC
6411	51	16 bit	SOCmin_M_abs	10	In percentuale di SOC
6411	52	16 bit	SOCmax_CS	50	In percentuale di SOC
6411	53	16 bit	SOCmin_CS	10	In percentuale di SOC
6411	54	16 bit	SOCmax_M_inj	95	In percentuale di SOC
6411	55	16 bit	SOCmin_M_inj	10	In percentuale di SOC
6411	56	16 bit	SOCmax_LS	95	In percentuale di SOC
6411	57	16 bit	SOCmin_LS	10	In percentuale di SOC
6411	58	16 bit	Pmax_Centralized_inj	1000	In W
6411	59	16 bit	Pmax_Centralized_abs	1000	In W
6411	60	16 bit	SOCmax_Centralized	95	In percentuale di SOC
6411	61	16 bit	SOCmin_Centralized	10	In percentuale di SOC

All’interno della tabella in cui sono indicati i parametri accessibili in lettura e scrittura sono presenti molti valori di tensioni, correnti, SOC e potenze che servono per caratterizzare il ruolo del sistema di accumulo all’interno della logica DBS. Oltre a questi è presente anche un parametro per la configurazione della modalità di funzionamento (DBS, centralizzata, Idle, etc.). Nella tabella per i valori in sola lettura invece si notano flag di stato, misure di tensioni e correnti erogate/assorbite dal sistema e bitmap di eventuali errori ad esso relativi.

Dopo aver definito la mappatura dei parametri si è passati all’implementazione del sistema di acquisizione e settaggio degli stessi. È stato scelto di implementare un sistema che permettesse di sfruttare gli SDO non solo per questa applicazione specifica ma anche in altre applicazioni. Questo in quanto l’implementazione di una soluzione generale avrebbe richiesto un tempo paragonabile all’implementazione dell’intero dizionario definito dalle tabelle precedenti che comunque avrebbe necessitato un input esterno per la definizione dei valori di configurazione. In primo luogo, è stato definito un formato del messaggio di configurazione che, tramite seriale,

va inviato al microcontrollore per poter utilizzare i messaggi SDO (Fig. 13).

Figura 13 Struttura del messaggio di configurazione da mandare tramite seriale SDO

Nella struttura del messaggio ogni elemento è suddiviso da un ";". Sono presenti, oltre ai valori che caratterizzano l'SDO che si vuole inviare, due indicatori di inizio e fine messaggio indicati con "SOF" ed "EOF" (Start Of Frame, End Of Frame).

Sulla base di questo formato è stato necessario costruire delle funzioni di verifica della conformità del messaggio e dei parametri inviati. Per esempio, è stato imposto che, nel caso il sistema sia in stato operazionale, gli SDO di tipo "write" vengano respinti. Inoltre, di default, la nanogrid si avvia in modalità di funzionamento e risponde con esito negativo ai messaggi di scrittura di SDO. Per poter utilizzare questa tipologia di messaggio è necessario, dunque, non solo che il sistema di accumulo non sia in stato operazionale ma anche che la nanogrid sia in modalità di configurazione, raggiungibile mediante la pressione di un determinato pulsante in fase di accensione. Così facendo si scongiura la possibilità di variazione dei parametri mentre i sistemi sono già avviati. Per poter usare questo sistema di configurazione e/o lettura è, ovviamente, necessario essere a conoscenza della mappatura (index, subindex, etc.) per poter settare i parametri in modo adeguato.

Oltre agli SDO non mancano ovviamente i messaggi di tipo PDO e NMT, mentre non vengono utilizzati, in questo caso, gli Heartbeat. Per quanto concerne l'unico PDO inviato dal sistema di accumulo esso ha come SID 0x180+nodeID, ha lunghezza 7 byte ed è stato mappato come indicato in Tab. 4.

Tabella 4 Mappatura PDO batteria a flusso

Campo	**Dato all' indice / Sottoindice della mappa di memoria**	**Dimensione**	**Nome**
0	6000 / 03	8 bit	Stato di Controllo
1	6000 / 01	8 bit	SOC Batteria
2	6000 / 02	8 bit	Flag Stato Runtime
3	6401 / 09	16 bit	Bitmap degli Allarmi presenti
4	6401 / 10	16 bit	Bitmap dei Blocchi

Anche questo PDO, come nel caso dell'accumulo al litio, viene letto ogni 1 ms poco prima dell'esecuzione dell'automa di logica assieme a quelli relativi alle batterie a litio. Ovviamente il tentativo di lettura e di decodifica di ogni PDO viene effettuato solo quando è presente il sistema di accumulo ad esso relativo, differentemente si procederà allla lettura dei messaggi per i sistemi presenti bypassando quelli non attivi.

1.4.3 Supercapacitore

Per quanto riguarda l'implementazione della comunicazione con i supercapacitori è stato scelto di muoversi similmente a quanto fatto con i sistemi di accumulo a flusso. Anche in questo caso, infatti, la comunicazione CANbus è cruciale in quanto serve per interfacciare non solo l'accumulo ma l'intero nodo, scambiando informazioni anche con il convertitore dei supercapacitori.

La scelta, effettuata in fase di implementazione della comunicazione con le batterie a flusso, di generalizzare l'utilizzo degli SDO è stata fatta anche in ottica dell'implementazione della comunicazione con i supercapacitori. Avendo la possibilità di utilizzare il messaggio seriale precedentemente descritto per usare gli SDO in fase sia di configurazione che di lettura, l'unica altra necessità è stata quella di implementare le routine di lettura dei messaggi PDO specifici del sistema a supercapacitori. Va evidenziato che, ovviamente, la struttura ipotizzata per l'utilizzo degli SDO ha comunque subito delle piccole modifiche per facilitarne l'utilizzo.

I supercapacitori utilizzano, dunque, SDO e PDO oltre, ovviamente, agli NMT. I PDO caratteristici di questo nodo, però, sono leggermente diversi da quelli descritti in precedenza in quanto si tratta di PDO configurati per essere inviati a seguito di richiesta RTR. Differentemente da quelli degli altri due sistemi i messaggi non vengono inviati continuamente ma solo a seguito di un messaggio di richiesta da parte del master. Per poter inviare il messaggio RTR è stato scelto di utilizzare un sistema simile a quello degli SDO, ovvero un messaggio tramite seriale con formato *SOF; RTR; EOF*. Una volta ricevuto il messaggio il sistema invierà una serie di PDO (in Tab. 5) che verranno letti con le stesse modalità di quelli dei sistemi precedenti.

Tabella 5 Mappatura PDO supercapacitore

Descrizione TxPDO	**COB**	**Node-id**	**PAR1**	**PAR2**	**PAR3**	**PAR4**
TxPDO1 (info supercap)	0x180	0x0n	SOC int16 in Wh	SOB int16 è 0 se bilanciamento SC ok	CURRENT SC int16 in A *10	TEMP SC int16 in C° *10 (media)
TxPDO2 (status)	0x280	0x0n	STATUSWORD SYS int16	STATUSWORD CONV int16	ERRCODE CRITIC int16	ERRCODE WARNING int16
TxPDO3 (derating / dontcare)	0x380	0x0n	%Potenza derating *10 int16	%Corrente derating *10 int16	Rif. Corrente dontcare in A *10 int16	DBS Status int16
TxPDO4 (info Convert.)	0x480	0x0n	POWER analog setpoint % *10 int16	TEMP CONV in C° *10 int16	VOLTAGE DCBUS int16 in V *10	CURRENT DCBUS int16 in A *10

Per quanto riguarda la mappatura degli SDO le tabelle risultano essere molto dispersive e per questo si è scelto di non inserirle. Va sottolineato che però è fondamentale conoscere la mappatura per poter impostare e leggere correttamente gli SDO.

1.5 Testing

1.5.1 Batteria a ioni di litio

Il primo passo ha visto il completamento dei test di comunicazione tra microcontrollori, aumentando la mole di dati da comunicare e integrando una libreria più articolata così da simulare parzialmente la comunicazione in CANOpen prevista per il sistema di accumulo. Questo ultimo passaggio non ha portato complicazioni ed è stata così conclusa la fase di test preliminare.

Il passo seguente è stato quello di azionare il sistema di storage e metterlo in comunicazione con il microcontrollore. Il sistema di accumulo preso in considerazione è uno storage a litio, più precisamente si tratta di un sistema a Litio-Ferro-Fosfato ($LiFePO_4$) formato da 4 moduli. Ogni modulo ha tensione nominale pari a 51.2 V, energia nominale 4.1 kWh, capacità nominale 80 Ah, peso 65 kg. Questi sono collegati tra loro in serie così da formare un rack complessivamente da 204.8 V e 16.4 kWh.

L'alimentazione della logica del sistema di accumulo prevede il collegamento di una batteria tampone da 12 V, attraverso questa si alimenta anche il BMS che valuta lo stato del sistema e gestisce la comunicazione tramite CANbus. La logica ha inoltre il controllo sui contattori che gestiscono la parte di potenza dell'accumulo. È stato necessario predisporre un setup di test collegando la logica del sistema di accumulo ad una batteria tampone di alimentazione, i morsetti di potenza ad un alimentatore (simulatore fotovoltaico) nel caso fosse necessario caricare lo storage e, in ultimo, formando un collegamento CANbus (CANH, CANL, GND) per la comunicazione tra il sistema ed il microcontrollore.

A questo punto è stato modificato il codice presente sul microcontrollore in modo tale da poter interagire con il BMS e quindi leggere o inviare messaggi. Non è stato necessario, dal punto di vista hardware, inserire le resistenze di terminazione sul bus di comunicazione perché già presenti nei transceiver del microcontrollore e dello storage. Va sottolineato che i due transceiver hanno tensioni di funzionamento diverse, quello del microcontrollore è a 3.3 V mentre quello del BMS è a 5 V. Questa differenza non presenta un problema in quanto, come già evidenziato nella prima fase dell'attività, i due transceiver sono compatibili.

Il bus di comunicazione è stato inizializzato impostando un bitrate di 125 kbps ed un sample time del 85-90%. Il BMS utilizza per la comunicazione PDO, HB e NMT. Rispetto a quelli che sono gli strumenti generalmente usati nel CANOpen, quindi, non sono usati gli SDO.

I messaggi NMT (Network management) servono per cambiare lo stato del sistema, inviando un NMT al BMS è possibile variare lo stato tra preoperational, stopped o operational. Il format per un NMT prevede di inviare 2 byte di dati contenenti sia lo stato che deve assumere il sistema sia l'indirizzo dello stesso (nel caso di bus a più nodi). Per quanto concerne il sistema di accumulo utilizzato, la ricezione di un NMT che varia lo stato in operational coincide con la chiusura del contattore per la gestione della potenza, viceversa una variazione da operational causa la riapertura del contattore e quindi la disconnessione del lato potenza del sistema.

Gli HB (Heartbeat) invece sono dei messaggi usati per monitorare lo stato del sistema di accumulo e sono inviati dal BMS con frequenza elevata per comunicare esclusivamente se si trova in

operational/stopped/preoperational.

I PDO sono messaggi inviati dal BMS del sistema di accumulo e contenenti i dati come tensioni, correnti, SOC, ecc., nel caso testato il BMS invia una serie di 8 PDO solo quando è in condizioni di operational.

Il microcontrollore è stato quindi programmato in funzione della mappatura dei PDO ed in modo tale che si possa scegliere se inviare gli NMT di operational e preoperational. È stato necessario impostare indirizzi ed indici per PDO e NMT ed inoltre implementare una funzione per la "decodifica" dei dati ricevuti e la stampa via seriale degli stessi così da visualizzare i valori letti. La decodifica dei PDO è necessaria in quanto un singolo PDO contiene più dati, ad esempio in un PDO di lunghezza 8 bytes i primi due possono essere dedicati alla tensione dello storage, un terzo per la corrente, uno per un contatore keep alive, ecc.

Una volta programmato il microcontrollore è stato collegato all'accumulo ed è stata alimentata la logica del BMS. Lo storage, come previsto, parte dallo stato preoperational. Il comportamento atteso alla ricezione del NMT operational è quello di chiudere i contattori ed indicare un SOC del 100% ed un setpoint di scarica di 0 A. questa incoerenza è dovuta all'attesa della carica massima per tarare il SOC, così da partire dal 100% e poi abilitare la scarica.

Conoscendo il comportamento atteso è stato quindi collegato un simulatore di fotovoltaico allo storage così da poterlo caricare. A questo punto è stato avviato il microcontrollore per testare la comunicazione. Come previsto lo storage ha ricevuto il primo NMT di operational ed ha chiuso i contattori inviando i PDO mappati, indicando SOC pari al 100% e setpoint di scarica pari a 0 A. È stato avviato quindi il simulatore per caricare il sistema e i PDO sono variati indicando la condizione di carica, la tensione, la corrente assorbita, ecc. Una volta raggiunto il completamento della carica è stato sbloccato il setpoint di scarica che ha cominciato ad indicare 80 A, mentre quello di carica è sceso a valori bassi prossimi a 0 A. Il test di comunicazione e carica è quindi stato completato con successo e si è passati ad effettuare altri due test, quelli di interruzione di comunicazione e quello di casistiche di warning/error/fault.

Il primo test è servito per verificare se fosse possibile rilevare un problema di comunicazione lato microcontrollore, ad esempio il caso in cui il cavo del CANbus venga danneggiato. Per far ciò è stato utilizzato il valore keep alive passato tramite PDO dal BMS. Questo è un contatore che varia ogni invio del PDO, la verifica prevede la valutazione di 3 keep alive consecutivi che, se di stesso valore, sono interpretati come un problema di comunicazione. Il risultato del test è stato positivo con il microcontrollore in grado di rilevare il distacco del cavo di comunicazione.

L'ultimo test ha visto l'analisi del PDO relativo agli errori, che è formato da una serie di bit flags ognuno dei quali relativo ad un errore, un allarme od un guasto. Il BMS, nel caso di error o fault, stacca il contattore lato potenza mentre, nel caso di warning, mantiene attivo lo scambio di potenza ma invia solamente il PDO di info sugli errori. Il test è stato effettuato causando un errore che provoca l'apertura del contattore, l'apertura della porta del rack di batterie. Come previsto all'apertura della porta è seguito il distacco del contattore con conseguente invio del PDO di info sugli errori ricevuto e decodificato dal microcontrollore. I test di comunicazione CANbus tra micro e storage sono stati perciò completati con successo.

Il passo seguente è stato quello più critico, la programmazione della comunicazione è stata integrata nel codice di gestione della nanogrid e sono stati effettuati dei test di funzionamento parallelo tra comunicazione e gestione dei convertitori.

Per prima cosa è stato scelto come gestire i tempi e come inserire la comunicazione nel codice. È stato scelto di leggere i PDO ricevuti in modo asincrono, più lentamente rispetto alla frequenza di invio del BMS, in quanto è stata ritenuta poco utile una lettura più rapida rispetto alla frequenza di verifica dei parametri da parte della logica della nanogrid. In seguito, sono stati scelti i dati effettivamente utili tra quelli inviati nei PDO e quali andassero utilizzati per la gestione o per il monitoraggio dello storage. I parametri ritenuti utili sono:

- SOC, ovvero la percentuale di carica residua, in funzione del quale si decide come può agire l'accumulo;
- V_batt ed I_batt, ovvero la tensione e la corrente dello storage che devono coincidere (con margine di errore piccolo) con quelle lette dai sensori di misura della nanogrid;
- Condizione di carica, un parametro che indica se la batteria è nello stato di carica/bilanciamento/carica completa;
- Setpoint di carica e scarica, mediante i quali si decide come limitare il comportamento in erogazione od assorbimento dell'accumulo;
- Keep alive, utile per verificare l'aggiornamento dei dati;
- Heartbeat, che serve per indicare lo stato dell'accumulo;
- Temperatura del rack, per monitorare la temperatura;
- Warning/error/fault.

Questa lista di parametri è modificabile andando ad aggiornare la lettura e decodifica dei PDO.

È stato in ultimo inserita nella nanogrid la possibilità di variare la configurazione in funzione della presenza o meno del BMS, dando quindi la possibilità di scegliere tra il lavorare con batterie al litio con BMS e comunicazione CANbus o con altri tipi di batterie non dotate di BMS.

A questo punto si è passati al test di comunicazione con la nanogrid mentre questa è in funzione. Questo test è stato fatto senza collegare il lato potenza dello storage e quindi con una nanogrid configurata per la gestione del convertitore di interfaccia con la rete elettrica e con il convertitore relativo ai carichi critici (UPS). Inoltre, è stato effettuato a posteriori rispetto ad un test di funzionamento della sola nanogrid (senza comunicazione) utile a verificare che l'integrazione della comunicazione non avesse compromesso la restante parte del codice.

Questo test ha portato numerosi inconvenienti. Dopo aver verificato il funzionamento della nanogrid senza il collegamento della comunicazione col BMS si è passati alla connessione del CANbus. È apparso subito evidente come il funzionamento venisse compromesso a causa del collegamento dei cavi di comunicazione del bus. I tentativi di capire la ragione di questo malfunzionamento hanno portato anche al danneggiamento di alcuni componenti ed alla relativa sostituzione che ha causato la perdita di tempo anche in virtù della necessità di effettuare nuovamente da capo tutti i test per individuare il problema. In ultimo è stata identificata la ragione del malfunzionamento nella mancanza di isolamento tra il microcontrollore ed il BMS che causava l'arrivo di numerosi disturbi sul microcontrollore e conseguentemente sui segnali di controllo dei convertitori. È stato quindi

necessario costruire una scheda con un transceiver isolato da utilizzare al posto del transceiver CAN integrato sulla scheda del microcontrollore.

Una volta applicata questa modifica il risultato ha portato al completamento del test ed al passaggio alla fase successiva: il collegamento del sistema di accumulo alla nanogrid.

È stato connesso il lato di potenza del sistema di accumulo alla nanogrid andando così ad aggiungere un altro convertitore oltre a quello di rete e UPS. Sono stati avviati diversi test sull'avvio della nanogrid a partire dal sistema di accumulo, sulla carica e la scarica e sulla connessione dello storage una volta che la nanogrid fosse già in funzionamento. Lo svolgimento di questi test è stato, ovviamente, molto lento e graduale in quanto, differentemente dalla comunicazione, in questo caso lo storage era direttamente collegato alla nanogrid e le potenze scambiate avrebbero potuto causare ingenti danni sia all'accumulo che al resto dei componenti.

I test hanno avuto esito positivo, concludendo quindi la sperimentazione della comunicazione CANbus nell'ottica della gestione dei sistemi di accumulo da parte della nanogrid.

A seguito di tali conclusioni è stato possibile impostare la struttura della comunicazione della nanogrid anche con altre tipologie di sistema di accumulo, come previsto da progetto. Si è passati quindi a definire le caratteristiche del protocollo da usare per l'interfacciamento con l'accumulo a supercapacitori.

Differentemente da quanto accade con lo storage a litio, i supercapacitori saranno gestiti da un convertitore non governato dal microcontrollore della nanogrid ma collegato con essa mediante CANbus per quanto riguarda la comunicazione, e mediante DC bus per quanto riguarda la potenza. Questo convertitore agirà separatamente e riceverà dal microcontrollore della nanogrid solamente le informazioni di configurazione, mentre invierà ad esso solo i dati ritenuti importanti per il monitoraggio dell'accumulo a supercapacitori.

Basandosi sull'esperienza pregressa sono state indicate le caratteristiche più importanti del bus da configurare, ovvero:

- utilizzo del CANOpen;
- bitrate di 125 kbps;
- integrabili sulla linea transceiver sia a 3.3 V che a 5 V necessariamente isolati;
- indirizzi del nodo master (micro nanogrid) e degli altri accumuli;
- negli 8 bytes (massimi) di un singolo messaggio RxPDO possono essere inserite diverse grandezze a condizione di fornire le opportune indicazioni sui contenuti così da "decifrare" i messaggi opportunamente;
- necessità di definire le grandezze utili/necessarie.

Partendo da questi punti cardine è iniziata la discussione con i vari partners per definire tutte le caratteristiche dell'accumulo a supercapacitori. Tra le varie proposte è stata accolta quella di utilizzare anche gli SDO precedentemente mai utilizzati, ma molto utili per la configurazione dei parametri logici del convertitore.

Sono stati previsti 6 SDO con codici unici per impostare i parametri necessari (es. soglie, delta di isteresi e la modalità di funzionamento MASTER/SLAVE/IDLE/DONTCARE) che sono ammissibili solamente mentre lo storage è in modalità preoperational in quanto la modifica di una singola soglia a caldo potrebbe provocare anomalie ed interferenze. Ovviamente il contenuto degli SDO è soggetto a verifica da parte del BMS dello storage

così da evitare errori di impostazione.

Oltre agli SDO sono previsti anche una serie di PDO strutturati come quelli del caso sperimentale ed ancora da definire in funzione del numero e natura delle grandezze che si sceglierà di inviare alla nanogrid.

Anche in questo caso è previsto l'invio da master (micro nanogrid) di un messaggio NMT per la gestione dello stato del BMS e per inviare il comando di passaggio in operational/preoperational.

È inoltre stato previsto un messaggio RTR inviato da master. Se un nodo riceve tale messaggio dovrà inviare i propri PDO corredati di node-id ed aggiornati.

In ultimo è stato previsto anche in questo caso un PDO relativo alle condizioni di emergenza che ancora vanno definite. Ancora in fase di definizione se sarà inviato anche un messaggio di servizio di natura emergency o solamente un PDO.

A questo punto, dopo aver quindi definito gran parte dei parametri di configurazione, il prossimo passo prevede il test della comunicazione tra la nanogrid (integrante già il sistema BMS delle batterie al litio) ed il prototipo di un dispositivo che sarà fornito prossimamente dai partners responsabili. Nella speranza di non riscontrare eccessivi problemi di comunicazione o di compatibilità (disturbi), mediante i test, si potranno limare i dettagli relativi alla gestione del sistema e perfezionare la costruzione di un prototipo di nanogrid integrante, oltre ai convertitori di rete e UPS, anche quelli relativi a storage di natura diversa (litio e supercapacitori, entrambi con BMS) ed a fonti di generazione rinnovabile (fotovoltaico).

1.5.2 Batteria a flusso

I test sulle batterie a flusso e sui supercapacitori non hanno, ovviamente, richiesto la fase di verifica del funzionamento del bus di comunicazione. Partendo dunque dai risultati ottenuti dai test con le batterie a litio il passo fondamentale è stata la validazione dell'implementazione della meccanica degli SDO per la gestione del convertitore relativo alle batterie a flusso.

Una volta effettuati gli opportuni collegamenti sulla scheda di simulazione per come già precedentemente descritto, è stato possibile avviare i test. Tramite l'interfaccia CAN bus e sfruttando un microcontrollore programmato per come descritto nel paragrafo precedente è stata verificata la possibilità di accedere ai parametri del sistema mediante SDO. La correttezza di questa e delle altre attività di test è stata verificata mediante l'interfaccia software sul PC precedentemente descritta.

Una volta verificata l'integrità della comunicazione il primo passo è stato effettuare la lettura dei parametri tramite SDO. Il software restituisce su seriale i valori letti nei vari SDO che vanno decodificati in funzione delle tabelle sopra riportate. Questo test ha avuto successo e si è passati ai test di scrittura. Mediante i medesimi passaggi, e cambiando solamente la tipologia di comando da read a write, sono stati impostati dei nuovi valori per la caratterizzazione del convertitore ed è stata anche verificata l'impossibilità di modifciare i valori dati solamente in lettura.

Arrivati a questo punto, dopo quindi essersi accertati del funzionamento della comunicazione e degli SDO, si è passati al check della macchina a stati del sistema come implementata in D2.1, Fig. 14.

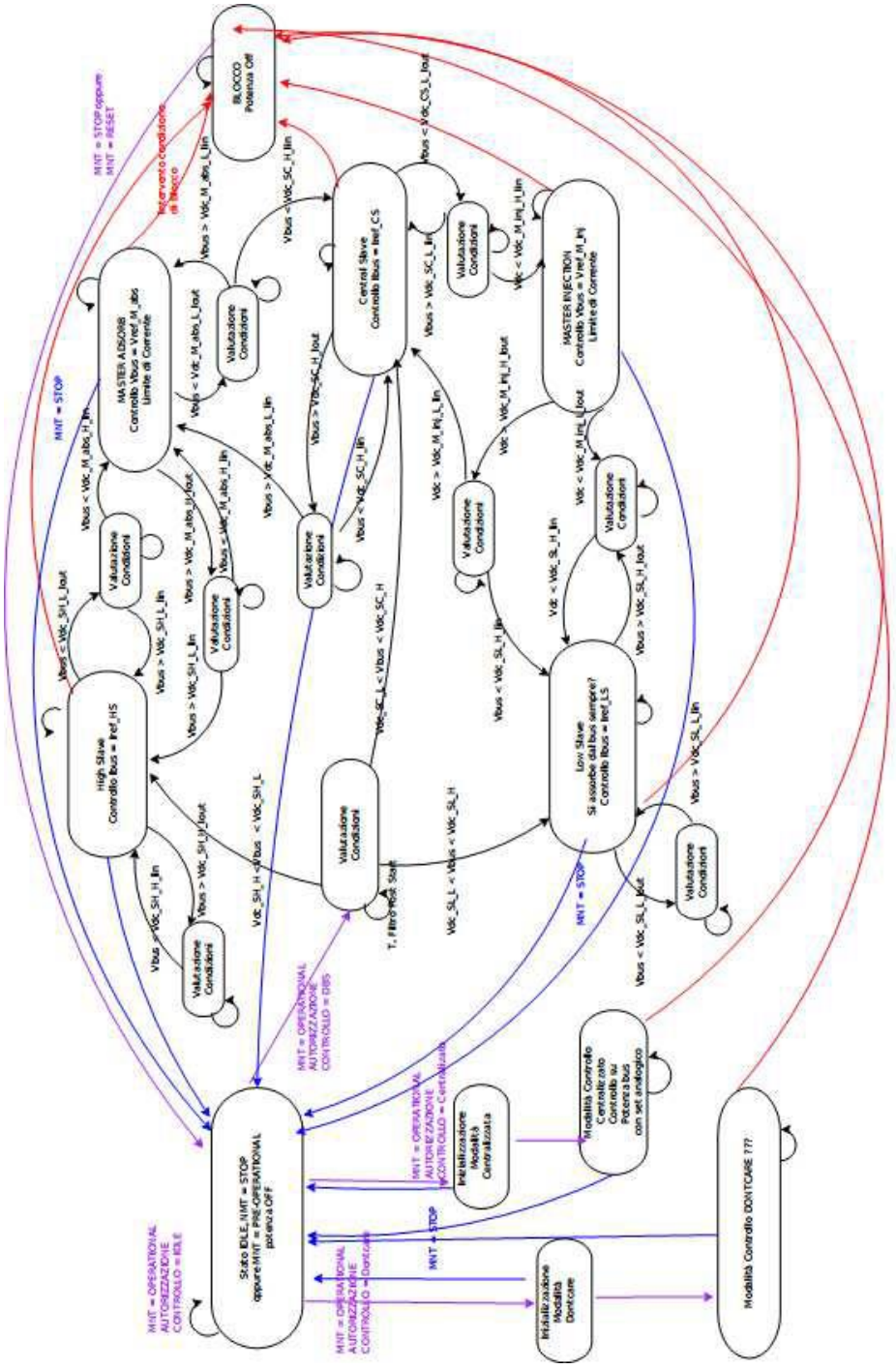


Figura 14 Macchina a stati batteria a flusso

La verifica è stata possibile grazie al software di simulazione utilizzato sul PC e connesso tramite un adattatore 485-USB. Mediante tale software sono stati fatti variare i parametri letti dal sistema (V_bus, corrente, tensione batteria, temperatura, etc.) e, tramite CAN bus, venivano letti i PDO inviati dal sistema che confermavano la movimentazione della macchina a stati.

Una volta terminati questi test è stata accertata la bontà del sistema di comunicazione e gestione sviluppato dal partner, per rinviare la verifica in potenza nell'attività A2.1 per come riportato in D2.1.

1.5.3 Supercapacitore

I test sul sistema a supercapacitori hanno seguito lo stesso iter di quelli relativi alle batterie a flusso, soprattutto in considerazione del fatto che la struttura è molto simile.

Si è partiti, dunque, dalla verifica del funzionamento della comunicazione con il sistema di gestione del convertitore. Anche in questo caso è stato collegato un sistema simulante l'hardware relativo al nodo dei supercap ad un software per il PC e, tramite CANbus, ad un microcontrollore programmato come descritto nel paragrafo relativo all'implementazione della comunicazione.

Mediante messaggi di tipo NMT è stato possibile muovere il sistema da operational e pre-operational e viceversa. Gli SDO inviati hanno sortito gli effetti attesi, anche se, in alcuni casi, è stato necessario riavviare il sistema per poter comunicare correttamente. Oltre ai messaggi SDO sono stati testate anche le altre tipologie di messaggio. Il messaggio PDO attivato dall'invio del messaggio RTR da parte del microcontrollore ha funzionato come previsto.

2 DEFINIZIONE DELL'INTERFACCIA DI COMUNICAZIONE SDM2 E TECNOLOGIE DI ACCUMULO NON CONVENZIONALE

A partire dalla definizione dei parametri da monitorare dei sistemi di accumulo non convenzionale (idrogeno e celle a combustibile, generatore alimentato a biodiesel, idraulico, termico, vehicle to grid (V2G)), è stato definito, progettato e realizzato un prototipo di sistema per acquisizione dati di misure elettriche ed energetiche per la gestione della nanogrid integrante una o più dei su menzionati sistemi di accumulo. A tale prototipo, nei disegni indicato come SCHEDA DI ACQUISIZIONE, è stato attribuito l'acronimo **SdM2**.

2.1 SCHEDA DI ACQUISIZIONE

2.1.1 Hardware

La scheda di acquisizione misure elettriche e termiche è basata su un processore Broadcom BCM2837B0, 64-bit SoC (System-on-a-Chip) quad core Cortex-A53 (ARMv8) @ 1.4 GHz. È dotata di una memoria da 1GB tipo LPDDR2 SDRAM.

Ha una connectivity basata su 2.4GHz e 5GHz IEEE 802.11.b/g/n/ac wireless LAN, Bluetooth 4.2, BLE, Gigabit Ethernet (maximum throughput 300Mbps), 4 porte USB 2.0. Supporto per Micro SD per caricare il sistema operativo e lo storage. Il sistema operativo caricato sulla SdM2 è Raspbian GNU/Linux 10 (buster) – Kernel 5.10.17 –v7+.

2.1.2 Firmware

Il linguaggio di programmazione utilizzato per lo sviluppo dell'applicazione è il **Python** 3.7.3.

Per consentire al gateway EUGENIO (apparato di congiunzione fra elementi del campo e la piattaforma

ComESto) di effettuare correttamente il "Discovering" dei dispositivi presenti nella rete interna è stato necessario adottare il protocollo di rete **mDNS** (Multicast DNS definito nella RFC6762). Esso consente la risoluzione di nomi in indirizzi IP senza la necessità della presenza di un server DNS all'interno della rete. Quando un dispositivo ha la necessità di conoscere l'indirizzo IP di un altro dispositivo in rete invia un pacchetto **UDP multicast** con la richiesta; il pacchetto, essendo *multicast*, raggiunge tutti i dispositivi connessi. La risposta viene inviata sempre in *multicast* in modo che tutti la ricevano. Di default, mDNS risolve nomi che terminano con il suffisso **.local**

Per poter utilizzare il protocollo mDNS lato SdM2, equipaggiato di sistema operativo Linux, bisogna assicurarsi che sia in funzione il <Daemon Avahi>. In particolare, sono stati creati due servizi per consentire ad EUGENIO di effettuare correttamente il discovering.

2.1.3 MQTT

Il protocollo base scelto per la comunicazione M2M (Machine-to-Machine) è MQTT. Esso fornisce un'alternativa alla tradizionale architettura Client/Server mediante un pattern **Publish/Subscribe**. Questo modello disaccoppia il client che invia il messaggio (il publisher) dal client o clients che ricevono il messaggio (i subscribers). I publishers ed i subscribers non si contattano mai l'un l'altro direttamente. Non sono nemmeno consapevoli che l'altro esiste. La connessione tra loro è gestita da un terzo componente, il **broker**. Il compito del broker è quello di filtrare tutti i messaggi in arrivo e distribuirli correttamente ai subscribers. Publishers e subscribers devono solo conoscere il nome host/IP e la porta del broker.

Oltre al disaccoppiamento spaziale, c'è un disaccoppiamento temporale: publishers e subscribers non devono essere eseguiti contemporaneamente. Il broker può archiviare messaggi per i subscribers che non sono online. Devono essere soddisfatte due condizioni per memorizzare i messaggi: il subscriber si è connesso con una sessione persistente e si è iscritto a un argomento con una **Quality of Service** (**QoS**) maggiore di 0. MQTT ha tre livelli di Quality of Service. È possibile specificare facilmente che un messaggio venga recapitato correttamente dal client al broker o dal broker a un client. Il QoS è un accordo tra il mittente di un messaggio e il destinatario di un messaggio e definisce la garanzia di consegna per un messaggio specifico. Ci sono 3 livelli di QoS in MQTT:

0 Al massimo una volta (At most once)

1 Almeno una volta (At least once)

2 Esattamente una volta (Exactly once)

Il client che pubblica il messaggio al broker definisce il livello di QoS del messaggio quando invia il messaggio al broker. Il broker trasmette questo messaggio ai client sottoscrittori utilizzando il livello QoS che ciascun client subscriber definisce durante il processo di subscribing. Se il client subscriber definisce una QoS inferiore rispetto al client di publish, il broker trasmette il messaggio con la qualità di servizio inferiore.

QoS è una caratteristica chiave del protocollo MQTT. QoS offre al client la possibilità di scegliere un livello di servizio che corrisponda all'affidabilità della rete e alla logica dell'applicazione. Poiché MQTT gestisce la

ritrasmissione dei messaggi e garantisce la consegna (anche quando il trasporto sottostante non è affidabile), QoS rende molto più semplice la comunicazione in reti inaffidabili.

Sussiste anche un disaccoppiamento della sincronizzazione: le operazioni su entrambi i componenti non devono essere interrotte durante il publishing o il receiving. MQTT funziona in modo asincrono. Poiché la maggior parte delle librerie client funziona in modo asincrono e si basa su callback o un modello simile, le attività non vengono bloccate durante l'attesa di un messaggio o la pubblicazione di un messaggio. In alcuni casi d'uso, la sincronizzazione è desiderabile e possibile. Per attendere un determinato messaggio, alcune librerie hanno API sincrone. Ma il flusso è solitamente asincrono.

Il meccanismo di Publish/Subscribe consente di scalare meglio del tradizionale approccio client-server. Questo perché le operazioni sul broker possono essere altamente parallelizzate e i messaggi possono essere elaborati in modo event-driven. La memorizzazione nella cache dei messaggi e l'instradamento intelligente dei messaggi sono spesso fattori decisivi per migliorare la scalabilità.

Il broker utilizza alcuni metodi per fare in modo che ogni subscriber riceva solo i messaggi a cui è interessato.

FILTRAGGIO IN BASE AL SOGGETTO

Questo filtro si basa sull'oggetto o sull'argomento che fa parte di ogni messaggio. Il cliente ricevente si iscrive al broker per argomenti di interesse. Da quel momento in poi, il broker assicura che il client ricevente ottenga tutti i messaggi pubblicati negli argomenti sottoscritti.

Ogni messaggio contiene un **topic** (oggetto) che il broker può utilizzare per determinare se un subscriber riceve o meno il messaggio.

In generale, gli argomenti sono stringhe con una struttura gerarchica che consente di filtrare in base a un numero limitato di espressioni.

FILTRO BASATO SUI CONTENUTI

Nel filtraggio basato sul contenuto, il broker filtra il messaggio in base a una specifica lingua di filtro del contenuto. I client riceventi si iscrivono per filtrare le query dei messaggi a cui sono interessati. Uno svantaggio significativo di questo metodo è che il contenuto del messaggio deve essere conosciuto in anticipo e non può essere crittografato o modificato facilmente.

FILTRAGGIO IN BASE AL TIPO

Quando vengono utilizzati linguaggi orientati agli oggetti, è una pratica comune utilizzare il filtraggio basato sul tipo/classe di un messaggio (evento).

CODE

È importante sottolineare che MQTT non ha nulla a che fare con una "coda di messaggi" tradizionale. Una coda di messaggi archivia i messaggi finché non vengono consumati. Quando si utilizza una coda di messaggi, ogni messaggio in arrivo viene archiviato nella coda finché non viene prelevato da un client (spesso chiamato

consumatore). Se nessun client preleva il messaggio, il messaggio rimane bloccato nella coda e attende di essere consumato.

Un messaggio viene consumato solo da un client. Un'altra grande differenza è che in una coda di messaggi tradizionale un messaggio può essere elaborato da un solo consumatore. In MQTT il comportamento è esattamente l'opposto: ogni subscribers che si iscrive al topic riceve il messaggio. Le code sono denominate e devono essere create in modo esplicito. Una coda è molto più rigida di un topic.

Prima di poter utilizzare una coda, questa deve essere creata esplicitamente con un comando separato. Solo dopo che la coda è stata nominata e creata è possibile pubblicare o consumare i messaggi. Al contrario, i topic MQTT sono estremamente flessibili e possono essere creati al volo.

2.1.3.1 *CLIENT*

Quando parliamo di un client, intendiamo un client MQTT. Sia i publishers che i subscribers sono client MQTT. Le etichette publisher e subscriber si riferiscono al fatto che il client stia attualmente pubblicando messaggi o si sia iscritto per ricevere messaggi (la funzionalità di pubblicazione e sottoscrizione può essere implementata anche nello stesso client MQTT). Un client MQTT è qualsiasi dispositivo (da un microcontroller fino a un server completo) che esegue una libreria MQTT e si connette a un broker MQTT su una rete. Fondamentalmente, qualsiasi dispositivo che parla MQTT su uno stack TCP/IP può essere chiamato client MQTT. L'implementazione client del protocollo MQTT è molto semplice e semplificata. La facilità di implementazione è uno dei motivi per cui MQTT è ideale per dispositivi di piccole dimensioni. Le librerie client MQTT sono disponibili per un'ampia varietà di linguaggi di programmazione.

2.1.3.2 *BROKER*

La controparte del client MQTT è il broker MQTT. Il broker è al centro di qualsiasi protocollo di publish/subscribe. A seconda dell'implementazione, un broker può gestire fino a milioni di client MQTT connessi contemporaneamente.

Il broker è responsabile della ricezione di tutti i messaggi, del filtraggio dei messaggi, della determinazione di chi è iscritto a ciascun messaggio e dell'invio del messaggio a questi client iscritti. Il broker conserva anche i dati di sessione di tutti i client che hanno sessioni persistenti, inclusi abbonamenti e messaggi persi. Un'altra responsabilità del broker è l'autenticazione e l'autorizzazione dei clienti. L'integrazione è particolarmente importante perché il broker è spesso il componente direttamente esposto su Internet, gestisce molti client e deve passare i messaggi ai sistemi di analisi ed elaborazione a valle. In breve, il broker è l'hub centrale attraverso il quale deve passare ogni messaggio. Pertanto, è importante che il tuo broker sia altamente scalabile, integrabile in sistemi di backend, facile da monitorare e (ovviamente) resistente ai guasti.

2.1.3.3 *CONNESSIONE MQTT*

Il protocollo MQTT è basato su TCP/IP. Sia il client che il broker devono disporre di uno stack TCP/IP. Una connessione MQTT è sempre tra un client e il broker. I client non si connettono mai direttamente tra loro. Per avviare una connessione, il client invia un messaggio CONNECT al broker. Il broker risponde con un

messaggio CONNACK e un codice di stato. Una volta stabilita la connessione, il broker la mantiene aperta finché il client non invia un comando di disconnessione o la connessione si interrompe.

In molti casi d'uso comuni, il client MQTT si trova dietro un router che utilizza la network address translation (NAT) per tradurre da un indirizzo di rete privato (come 192.168.x.x, 10.0.x.x) a un indirizzo pubblico. Come già accennato, il client MQTT avvia la connessione inviando un messaggio CONNECT al broker. Poiché il broker ha un indirizzo pubblico e mantiene la connessione aperta per consentire l'invio e la ricezione bidirezionale di messaggi (dopo il CONNECT iniziale), non c'è alcun problema con i client che si trovano dietro un NAT.

Alcune informazioni incluse in un messaggio CONNECT sono:

- ***ClientId*** L'identificatore del client (ClientId) identifica ogni client MQTT che si connette a un broker MQTT. Il broker utilizza il ClientId per identificare il client e lo stato corrente del client. Pertanto, questo Id dovrebbe essere univoco per client e broker. Il ClientId vuoto risulta in una connessione senza alcuno stato. In questo caso, il flag di Clean session deve essere impostato su true o il broker rifiuterà la connessione.
- ***Clean Session*** Il flag di Clean session indica al broker se il client desidera stabilire una sessione persistente o meno. In una sessione persistente (CleanSession = false), il broker memorizza tutte le subscriptions per il client e tutti i messaggi persi per il client che ha sottoscritto un livello di qualità del servizio (QoS) 1 o 2. Se la sessione non è persistente (CleanSession = true), il broker non memorizza nulla per il cliente ed elimina tutte le informazioni da qualsiasi precedente sessione persistente.
- ***Username/Password*** MQTT può inviare un nome utente e una password per l'autenticazione e l'autorizzazione del client. Tuttavia, se queste informazioni non sono crittografate o con hash (mediante implementazione o TLS), la password viene inviata in testo normale. Alcuni broker possono autenticare i client con un certificato SSL, quindi, non sono necessari nome utente e password.
- ***Will Message*** L'ultimo Will Message fa parte della funzione Last Will and Testament (LWT) di MQTT. Questo messaggio notifica ad altri client quando un client si disconnette in modo insolito. Quando un client si connette, può fornire al broker un 'last will' sotto forma di messaggio MQTT e topic all'interno del messaggio CONNECT. Se il client si disconnette in modo insolito, il broker invia il messaggio LWT per conto del client.
- ***KeepAlive*** Il KeepAlive è un intervallo di tempo in secondi che il client specifica e comunica al broker quando viene stabilita la connessione. Questo intervallo definisce il periodo di tempo più lungo che il broker e il cliente possono sopportare senza inviare un messaggio. Il client si impegna a inviare messaggi di richiesta PING regolari al broker. Il broker risponde con una risposta PING. Questo metodo consente a entrambe le parti di determinare se l'altra è ancora disponibile.

Quando un broker riceve un messaggio CONNECT, è obbligato a rispondere con un messaggio CONNACK.
Il messaggio CONNACK contiene due voci di dati:

- *Flag* di sessione presente

- Un return *Code* della connessione

Flag di sessione presente

Il flag di sessione presente indica al client se il broker ha già una sessione persistente disponibile dalle precedenti interazioni con il client. Quando un client si connette con Clean Session impostato su true, il flag di sessione presente è sempre falso perché non è disponibile alcuna sessione. Se un client si connette con Clean Session impostato su false, ci sono due possibilità: se le informazioni sulla sessione sono disponibili per ClientId e il broker ha memorizzato le informazioni sulla sessione, il flag di sessione presente è true. Altrimenti, se il broker non dispone di informazioni sulla sessione per il ClientId, il flag di presenza della sessione è false. Questo flag è stato aggiunto per aiutare i client a determinare se devono sottoscrivere topics o se i topics sono ancora archiviati in una sessione permanente.

Return Code della connessione

Il secondo flag nel messaggio CONNACK è il flag di riconoscimento connessione. Questo flag contiene un codice di ritorno che indica al client se il tentativo di connessione è riuscito o meno.

Return Code	Return Code response
0	Connection accepted
1	Connection refused, unacceptable protocol version
2	Connection refused, identifier rejected
3	Connection refused, server unavailable
4	Connection refused, bad user name or password
5	Connection refused, not authorized

2.1.4 INSTALLAZIONE SOFTWARE/LIBRARY

Per implementare il protocollo MQTT, utilizzato per lo scambio di messaggi fra Publisher e Subscriber, è stato installato il software Mosquitto. La versione utilizzata è la Mosquitto **MQTT v3.1/3.1.1** Broker. È stata installata Paho MQTT Python client library. Questo codice fornisce una classe '*client*' che consente alle applicazioni di connettersi ad un broker MQTT per pubblicare messaggi, subscribe topics e ricevere messaggi pubblicati. Fornisce anche alcune funzioni di supporto per rendere molto semplice la pubblicazione di messaggi su un server MQTT.

- Installato un client **OPC-UA**;
- Versione: opcua-0.98.13 opcua-client-0.8.0 opcua-widgets-0.5.10 pytz-2021.1;
- Installato **CANopen 1.2.1.**

2.1.5 STRUTTURA JSON

I dati scambiati con il gateway EUGENIO, il cui utilizzo è indicato in dettaglio a partire dal paragrafo 3.1, vengono incapsulati in un messaggio di tipo stringa codificato secondo il formato JSON (JavaScript Object Notation). Il messaggio è suddiviso in una sezione denominata HEADER ed in una sezione denominata BODY. Nella header vengono inserite le informazioni per il controllo del flusso (numero sequenziale, ID della richiesta, ecc.). In particolare, va sempre inserita la chiave "**operation**" che indica il tipo di operazione a cui si riferisce il BODY del messaggio. La chiave operation può essere di tipo *action* o *state*, a seconda che sia utilizzata per "inviare richieste ai partecipanti" o inviare spontaneamente informazioni ai partecipanti. Nel body del messaggio vanno inseriti tutti gli altri parametri. I dettagli sulla struttura dei JSON creati e sullo schema adottato per la definizione di tutti i parametri che ne fanno parte, sono descritti nel documento [1] riportato nei riferimenti.
Le richieste da parte di EUGENIO arrivano sul canale ng/operations; ad essi bisogna dare una risposta nei formati definiti. I messaggi contenenti valori misurati e che vengono inviati spontaneamente utilizzeranno il canale ng/state.

3 CONFIGURAZIONE DEL SISTEMA DI ACQUISIZIONE PER SISTEMA DI ACCUMULO NON CONVENZIONALE

Per l'implementazione del prototipo di SdM2, in vista della configurazione del dimostratore, oggetto di sviluppo dell'attività A7.2, sono stati presi in considerazione i generatori biodiesel (BIODIESEL e GRUPPO ELETTROGENO) e l'accumulo mediante IDROGENO. Nella Fig. 15 è raffigurato l'insieme delle apparecchiature implicate, nonché le tipologie di connessione adoperate per la realizzazione del sistema SdM2. Nella Fig. 16 è riportato il particolare della scheda di acquisizione.

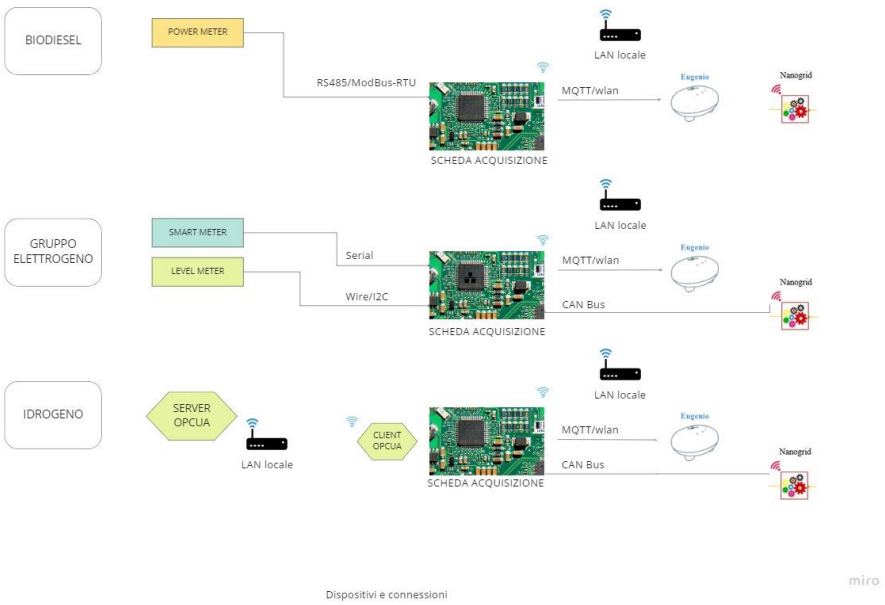


Figura 15 Architettura dei dispositivi e connessione per il sistema SdM2 per idrogeno e Biodiesel

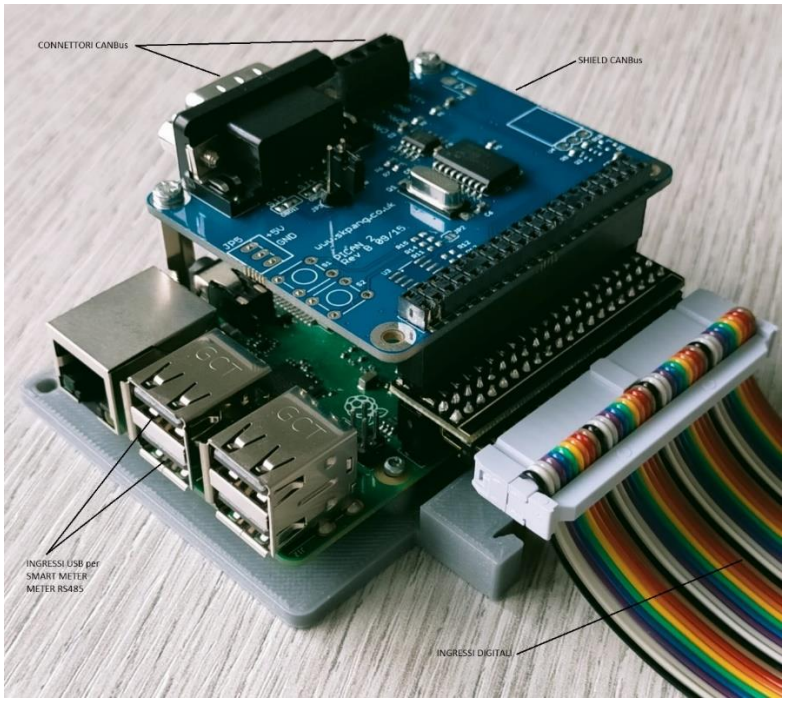


Figura 16 Particolare della scheda di acquisizione

3.1 BIODIESEL

Il primo accumulo non convenzionale preso in considerazione riguarda la produzione di biodiesel e l'uso di questo per alimentare un gruppo elettrogeno. È previsto l'utilizzo di un SINGLE PHASE MULTI-FUNCTION ENERGY METER dotato di RS485 Modbus communication ed in grado di misurare kW, kWh, V, A, FP, Hz. Le sue caratteristiche sono riportate nell'ALLEGATO 1. I carichi elettrici delle apparecchiature necessarie alla produzione di biodiesel (riscaldatori, agitatore, ecc.) vengono alimentati da una rete elettrica che fa capo al suddetto POWER METER. I dati prelevati dalla uscita RS485, convogliati in un convertitore RS485/USB vengono inviati all'interfaccia relativa della SdM2.

I dati vengono quindi incapsulati in un messaggio di tipo stringa codificato secondo il formato JSON (JavaScript Object Notation) e scambiati con EUGENIO tramite MQTT. Nella header del contenuto del messaggio vengono inserite delle informazioni per il controllo del flusso. In particolare, va sempre inserita la chiave "**operation**" che indica il tipo di operazione a cui si riferisce il corpo del messaggio.

Oltre a ricevere richieste da parte di EUGENIO (canale **ng/operations**) a cui dare risposta, ci sono messaggi contenenti valori misurati che vengono inviate spontaneamente da SdM2 (canale **ng/state**).

È stato stabilito che la frequenza di invio verso EUGENIO di questi messaggi è di **10** minuti.

Di seguito le definizioni utilizzate negli invii mediante MQTT ad EUGENIO:

System_id	System	System_type

80	biodiesel_sub_system	bio_storage

In Tab. 6 sono riportati i dati misurati dal sistema di accumulo Biodiesel.

Tabella 6 Dati misurati accumulo Biodiesel

DATI MISURATI – Accumulo BIODIESEL		
Definizione	**Unità di misura**	**ID Parametro**
Anomalia accumulo	1/0	50
Disponibilità accumulo	1/0	51
Stima dell'energia accumulabile	kWh_{eq}	52
Potenza elettrica assorbita	kW	53
Profilo di potenza assorbita	kW	54
SOCist	%	55

3.2 GRUPPO ELETTROGENO

Il secondo accumulo non convenzionale preso in considerazione, nell'ambito dei generatori biodiesel, riguarda la generazione di energia elettrica mediante gruppo elettrogeno ed utilizzando come carburante il biodiesel.

È prevista la misura di energia elettrica prodotta sia monofase che trifase. Per entrambe si utilizzeranno due SMART METER configurati opportunamente. Ognuno è dotato di trasformatori amperometrici (TA). Il collegamento con SdM2 avviene mediante connessione seriale con connettore USB.

Le caratteristiche tecniche sono riportate in ALLEGATO 2. Prima del suo utilizzo, lo SMART METER deve essere configurato seguendo le indicazioni contenute nel Manuale di installazione Smart Meter che viene fornito a corredo dell'apparecchiatura e distinguendo tra configurazione monofase e configurazione trifase. Il dispositivo può essere fissato su barra DIN, ma dispone di appositi fori nella parte posteriore che consentono di fissarlo su una parete.

Lo SMART METER fornisce in uscita una stringa dalla quale è possibile estrapolare i valori misurati. Un esempio di stringa è riportato qui di seguito.

p1=xx.x & p2=xx.x & p3=xx.x & p4=xx.x & p5=xx.x & p6=xx.x & e1=xx.x & e2=xx.x & e3=xx.x & e4=xx.x & e5=xx.x & e6=xx.x & en1=xx.x & en2=xx.x & en3=xx.x & en4=xx.x & en5=xx.x & en6=xx.x

p1 rappresenta la potenza misurata del canale 1 espressa in W, e1 è l'energia misurata del canale 1 in Wh, en1 rappresenta l'energia negata del canale 1. I valori negativi sono riferiti a potenze/energie immesse in rete.

Oltre allo SMART METER viene utilizzato un LEVEL METER, progettato ad hoc per rilevare il livello di carburante presente nel serbatoio del gruppo elettrogeno.

Di seguito le definizioni utilizzate negli invii mediante MQTT ad EUGENIO:

System_id	System	System_type
81	biodiesel_sub_system	bio_production

In Tab. 7 sono riportati i dati misurati dal sistema Gruppo Elettrogeno.

Tabella 7 Dati misurati Gruppo Elettrogeno

DATI MISURATI – Gruppo Elettrogeno		
Definizione	**Unità di misura**	**ID Parametro**
Anomalia generazione	1/0	60
Disponibilità erogazione	1/0	61
Stima dell'energia erogabile	kWh_{eq}	62
Potenza elettrica generata	kW	63
Profilo di potenza generata	kW	64
SOCist	%	65
Anomalia di sistema generatore Biodiesel + gruppo elettrogeno	1/0	66

Anche qui i dati vengono quindi incapsulati in un messaggio di tipo stringa codificato secondo il formato JSON (JavaScript Object Notation) e scambiati con EUGENIO tramite MQTT. Nella header del contenuto del messaggio vengono inserite delle informazioni per il controllo del flusso. In particolare, va sempre inserita la chiave "operation" che indica il tipo di operazione a cui si riferisce il corpo del messaggio.

Oltre allo scambio dati con EUGENIO, è previsto di inviare le informazioni anche verso il controllore della nanogrid. Questo scambio avverrà mediante invio su **CANbus**.

Il settaggio delle caratteristiche del bus CAN di comunicazione prevede essenzialmente:

- CAN 2.0 A con identificativo a 11 bit
- Bitrate 125 kbps

La SdM2 invia solamente dei dati spontaneamente e non ha un controller che deve essere gestito dalla nanogrid. Ricordando che tra le caratteristiche configurabili nelle librerie del **CANOpen** ci sono i messaggi che vengono usati lo scambio di dati e per la gestione della comunicazione, in questo caso si è pensato di utilizzare soltanto dei PDO (Process Data Object), la cui struttura prevede:

CAN-ID	RTR	LEN	DATA BYTES

Con:

CAN-ID: rappresenta l'ID del messaggio;

RTR: remote transmission request, utilizzabile solamente con i messaggi configurati per accettare richiesta di invio dal master;

LEN: indica la lunghezza del messaggio, che può arrivare fino al massimo di 8 byte;

DATA BYTES: byte contenenti i dati da inviare.

Il microcontrollore di gestione e controllo viene programmato per poter decodificare i dati ricevuti mediante PDO così strutturati, nonché altre tipologie di messaggi. La decodifica dei PDO è necessaria in quanto un singolo PDO contiene più dati distribuiti sui byte a disposizione. Pertanto, in un singolo messaggio PDO potranno essere "accorpate" e trasferite più grandezze acquisite dal sistema.

La struttura dei TPDO (PDO trasmessi da SdM2) utilizzati in questo accumulo non convenzionale è la seguente:

GRUPPO ELETTROGENO **node-id** = 0xXX

CAN-ID	RTR	DLC	B0	B1	B2	B3	B4	B5	B6	B7
180h + node-id	0	8	Stima dell'energia erogabile kWheq * 10		Potenza elettrica generata kW * 10		SOCist %	0	0	0
280h + node-id	0	8	bit 0 Anomalia erogazione		0	0	0	0	0	0
			bit 1 Disponibilità erogazione							
			bit 2 Anomalia di sistema generazione biodiesel+gruppo elettrogeno							

oltre ad un timestamp in formato epoch.

3.2.1 MODULO RILEVAZIONE LIVELLO

3.2.1.1 *PREMESSA*

Nell'accumulo biodiesel è presente un gruppo elettrogeno fornito dall'azienda MOSA. Il modello adottato è GE 15 YSXC ed è dotato di un indicatore del livello di carburante presente nel serbatoio di tipo analogico.

Figura 17 Indicatore di livello di carburante

La lettura viene fatta su uno strumento con indicatore a lancetta della AUTEL, visibile in Fig. 17 (la specifica è riportata in appendice come allegato A).

Il sensore di livello all'interno del serbatoio è di tipo elettromagnetico resistivo e fornisce in uscita un segnale continuo. Il sensore contiene una catena di contatti reed distanziati uno dall'altro. Il magnete posto nel galleggiante chiude in successione i contatti reed posti nello stelo collegando l'uscita ad un punto via via diverso della catena di resistenze.

La resistenza in uscita dal sensore aumenta al diminuire del livello di carburante presente nel serbatoio.

La resistenza del singolo passo viene determinata in base alle richieste del cliente e della lunghezza del sensore.

Questa informazione è rilevabile dalla curva richiesta all' azienda MOSA riportata in Fig. 18.

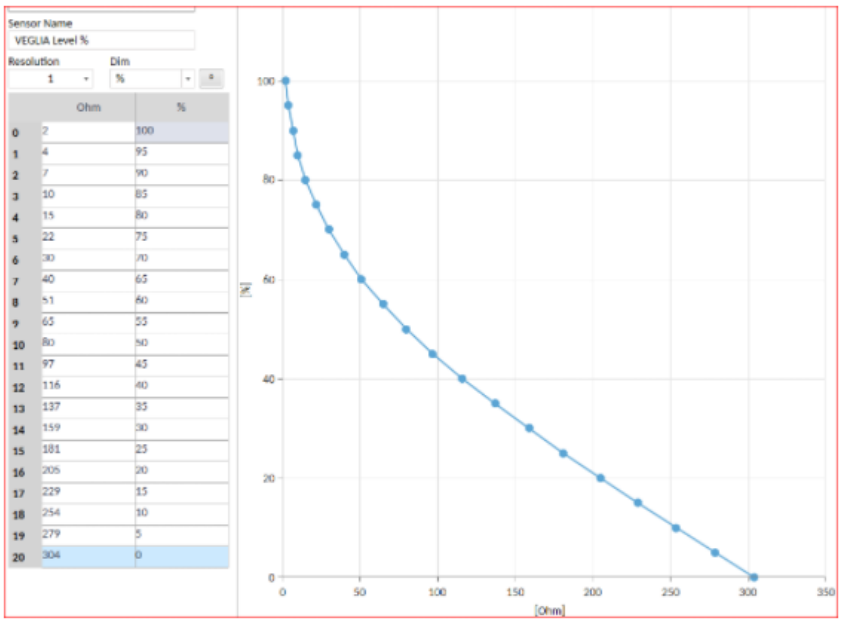


Figura 18 Curva di rilevazione resistenza in uscita -livello di carburante

3.2.1.2 *REALIZZAZIONE INTERFACCIA LIVELLO*

Per poter rilevare la quantità di biodiesel presente nel serbatoio carburante del gruppo elettrogeno si è pensato di utilizzare lo stesso segnale che viene inviato all'indicatore di livello presente sul gruppo. Pertanto, tramite lo schema elettrico della macchina, è stato individuato il punto da cui prendere il segnale. Tale segnale è rappresentato dalla tensione elettrica continua ***VR*** presente ai capi della resistenza (variabile con il livello di carburante) del sensore di livello.

3.2.1.2.1 RILIEVI

Come primo step sono state condotte delle misurazioni sul gruppo elettrogeno. In particolare, è stata rilevata una curva:

Carburante presente nel serbatoio/Tensione ai capi del sensore di livello

Il serbatoio è stato svuotato e sono state immesse via via delle quantità prestabilite di carburante. Ad ogni aggiunta di carburante è stata misurata la **VR** sia a gruppo spento che a gruppo acceso. È stata misurata anche la tensione di alimentazione dell'indicatore a lancetta.

Risulta che la tensione di alimentazione ***VA*** rilevata sull'indicatore di livello è pari a:

VA = 11.9 Volt a gruppo elettrogeno spento (il gruppo è dotato di batteria)

VA = 14.3 Volt a gruppo elettrogeno acceso

La tensione VA verrà utilizzata per alimentare la scheda di interfaccia realizzata.

I livelli di tensione rilevati di **VR** vanno da 6 Volt a serbatoio vuoto, a 0.55 Volt a serbatoio pieno (60 lt)

Pertanto, è stato necessario considerare la necessaria traslazione dei livelli di tensione per rispettare quelli di funzionamento del microcontrollore utilizzato per la scheda di acquisizione, pari **3.3** Volt.

È stato tenuto in considerazione anche il livello minimo di carburante all'interno del serbatoio che è pari a 6 lt.

3.2.1.2.2 CURVE

Nella Fig. 19 si riporta lo schema delle connessioni fra le parti implicate nella rilevazione del livello di carburante presente nel serbatoio del gruppo elettrogeno.

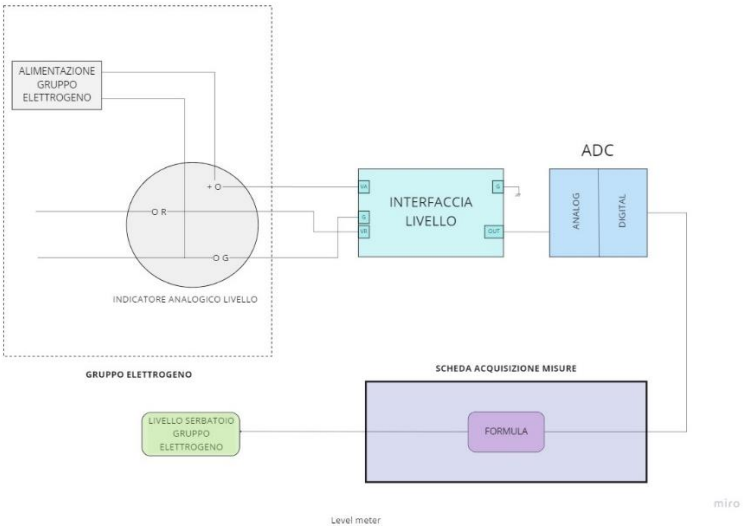


Figura 19 Schema di connessione del livello di carburante

Nel riquadro "GRUPPO ELETTROGENO" è presente l'alimentazione del gruppo elettrogeno, dal quale viene derivata la tensione di alimentazione della INTERFACCIA LIVELLO. L'indicatore analogico livello rappresenta lo strumento della AUTEL. In particolare, è visibile il punto indicato con **R**, dove arriva il segnale proveniente dal sensore di livello.

Il punto indicato con **VR** sulla INTERFACCIA LIVELLO è ad alta impedenza per non influire sulla misura dell'indicatore analogico di livello. Valori rilevati sulla INTERFACCIA LIVELLO indicano correnti assorbite dell'ordine di qualche µA quando la **VR** è massima. Con alimentazione dell'interfaccia a 14.3 Volt l'assorbimento della scheda è circa 7 mA.

La Fig. 20 mostra l'andamento della tensione di uscita ***Vout*** della INTERFACCIA LIVELLO al variare della tensione in ingresso **VR**. L'andamento mostra una relazione lineare tra le due grandezze.

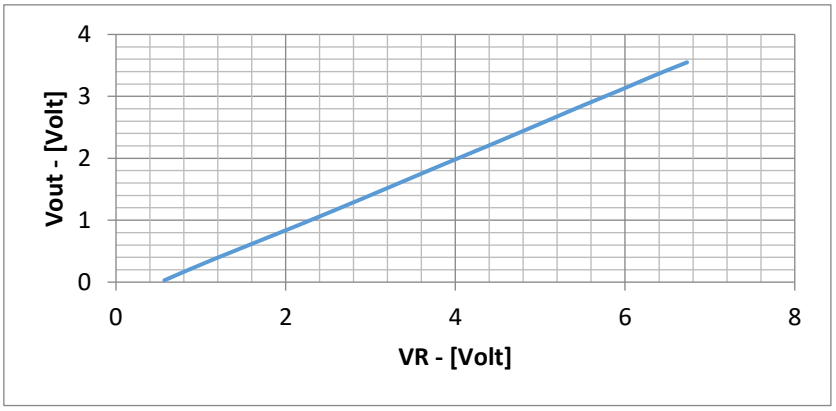


Figura 20 Andamento Vout-Vr

I valori sono stati ottenuti alimentando la scheda con una tensione fissa di 14.3 Volt ed utilizzando un alimentatore variabile per ottenere dei punti prestabiliti per la tensione ***VR***.

La tensione di uscita della INTERFACCIA LIVELLO ***Vout*** viene inviata ad un low-power 16 bit converter **ADC** che incorpora un amplificatore programmabile ed un comparatore digitale.

In Fig. 21 è riportato un grafico che include l'andamento della tensione VR misurata sul gruppo elettrogeno, traslata di livello e messa in relazione ai litri contenuti nel serbatoio, l'uscita misurata sulla scheda reale avente come ingresso la tensione VR, sempre in relazione ai litri contenuti nel serbatoio e la curva Poli, che è la rappresentazione della curva di tendenza con la corrispondente formula matematica.

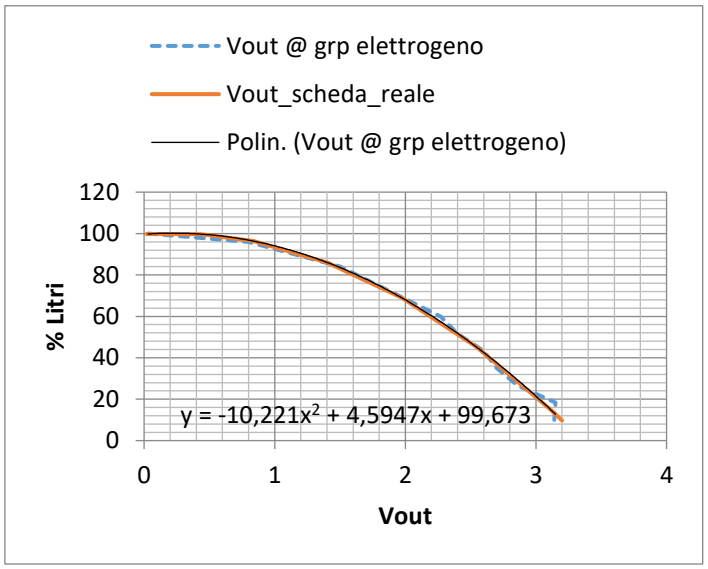


Figura 21 Andamento Vout-litri

Il segnale digitale in uscita dell'**ADC** è inviato alla scheda di acquisizione misure **SdM2**. Nel FW di gestione della scheda è presente una funzione che a partire dal valore di tensione ***Vout*** ricava la percentuale di litri di carburante presenti nel serbatoio del gruppo elettrogeno.

3.2.1.2.3 RILEVAZIONE ANOMALIA GRUPPO ELETTROGENO

Non avendo a disposizione segnali digitali da poter prelevare dal circuito del gruppo elettrogeno, per rilevare anomalie che potrebbe presentare il gruppo si è pensato di sfruttare la presenza di un diodo led di segnalazione anomalia presente sul pannello del gruppo. Per ottenere questa segnalazione verrà prelevata la tensione ai capi del led di segnalazione ottenendo un valore di tensione che ricade nel range 1.3 – 1.5 Volt quando il led è attivo ed è praticamente nullo quando il led è spento. Tale segnale viene inviato all'ingresso di un amplificatore differenziale presente all'interno dell'ADC utilizzato per la misura del livello di carburante presente nel serbatoio.

3.3 IDROGENO & FUEL-CELL

Il terzo accumulo non convenzionale preso in considerazione riguarda il sistema di accumulo ad idrogeno e celle combustibili. L'eccesso di energia prodotta dagli impianti a fonte rinnovabile viene accumulata tramite

un elettrolizzatore per la produzione di idrogeno da stoccare all'interno di serbatoi. L'idrogeno accumulato potrà essere utilizzato in differita come combustibile della cella che produce elettricità (generatore PEM).

Il sistema di gestione e controllo dell'impianto comunica i dati relativi al funzionamento mediante protocollo standard **OPC-UA** (Open Platform Communications Unified Architecture). L'interfaccia di comunicazione dovrà recepire tali dati ed inviarli sul CAN bus al controller della Nanogrid.

In particolare, il sistema di gestione dell'impianto è dotato di un server OPC-UA e pertanto la SCHEDA DI ACQUISIZIONE SdM2 sarà dotata di CLIENT per dialogare tramite il suddetto protocollo. Il dialogo avverrà tramite rete WiFi. Naturalmente ci saranno dei parametri di comunicazione che dovranno essere recepiti nella configurazione del client OPC-UA e resi disponibili dal server (vedi paragrafo dedicato al testing).

Oltre ad essere inviati su CANbus, i dati vengono incapsulati in un messaggio di tipo stringa codificato secondo il formato JSON (JavaScript Object Notation) ed inviati verso EUGENIO tramite MQTT.

Di seguito le definizioni utilizzate negli invii mediante MQTT ad EUGENIO:

System_id	System	System_type
80	fuel_cell_sub_system	fuel_cell

In Tab. 8 sono riportati i dati misurati dal sistema Gruppo Elettrogeno.

Tabella 8 Dati misurati accumulo ad idrogeno

DATI MISURATI – Accumulo IDROGENO & FUEL-CELL		
Definizione	**Unità di misura**	**ID Parametro**
Richiesta accumulo	1/0	70
Richiesta erogazione	1/0	71
Anomalia accumulo	1/0	72
Anomalia generazione	1/0	73
Stima dell'energia erogabile	kWh_{eq}	74
Disponibilità accumulo	1/0	75
Disponibilità erogazione	1/0	76
Anomalia sistema	1/0	77
Potenza elettrica assorbita	kW	78
Potenza elettrica generata	kW	79
Stima dell'energia accumulabile	kWheq	80
SOCist	%	81
Efficienza istantanea del sistema di accumulo	number	82
Efficienza istantanea del sistema di erogazione	number	83
Potenza istantanea auto-consumata in carica	number	84
Potenza istantanea auto-consumata in scarica	number	85

Per quanto concerne l'invio dei dati su **CANbus**, la struttura dei TPDO (PDO trasmessi da SdM2) utilizzati in questo accumulo non convenzionale è la seguente:

IDROGENO **node-id** = 0xYY

CAN-ID	RTR	DLC	B0	B1	B2	B3	B4	B5	B6	B7
180h + node-id	0	8	Stima dell'energia erogabile kWheq * 10		Potenza elettrica assorbita kW * 10		Potenza elettrica generata kW * 10		Stima dell'energia accumulabile kWheq * 10	
280h + node-id	0	8	Efficienza istantanea del sistema di accumulo Valore * 10		Efficienza istantanea del sistema di erogazione Valore * 10		Potenza istantanea auto-consumata in carica Valore * 10		Potenza istantanea auto-consumata in scarica Valore * 10	
380h + node-id	0	8	bit 0 Richiesta accumulo bit 1 Richiesta erogazione bit 2 Anomalia accumulo bit 3 Anomalia generazione bit 4 Disponibilità accumulo bit 5 Disponibilità erogazione bit 6 Anomalia sistema Bit 7 a bit 15 = 0		0	0	SOCist %	0	0	0

4 TESTING

4.1 SELEZIONE HW

Nella fase iniziale dello sviluppo del FW si è utilizzata una scheda con processore ARM Cortex-M3 @ 84 MHz dotata di due CAN controller. Sono stati effettuati dei test su CAN BUS utilizzando transceiver esterni alimentati a 3.3V. Come linguaggio di programmazione si è utilizzato C++. È stata sviluppata una macchina a stati per ottemperare a specifiche CANOpen. Di seguito è stata introdotta una scheda con processore Cortex-A53 e come linguaggio di programmazione è stato utilizzato il **Python 3.7.3**. Tramite la scheda con processore A53 è possibile controllare la scheda con processore ARM Cortex-M3, dando così al sistema una maggiore flessibilità ed una più ampia gamma di interfacce a disposizione.

Tuttavia, ottimizzando lo sviluppo, la scelta definitiva è ricaduta sulla scheda con processore A53. Ciò consente una maggiore affidabilità al costo di qualche componente aggiuntivo per sostituire delle funzionalità della scheda con processore ARM Cortex-M3.

4.2 SERVER OPC-UA

L'impianto di produzione ed accumulo dell'idrogeno sarà dotato di un PLC di gestione e controllo con server OPC-UA. L'interfaccia SdM2 dovrà acquisire le informazioni rese disponibili dal server ed inviarle su CAN bus. Pertanto, nella SCHEDA DI ACQUISIZIONE si è implementato un client OPC-UA in grado di dialogare con tale server. In mancanza di un'apparecchiatura reale, si è implementato un SERVER OPC-UA scritto utilizzando il linguaggio ***Python*** ed installato su una macchina Windows 10 per poter simulare la situazione reale.

Il collegamento client-server è stato realizzato con rete wifi perché si presuppone che nella realtà si userà la stessa tecnologia. Per essere quanto più vicini alla situazione reale sono stati considerati tutti i segnali presenti nella Tab. 8 e sono stati inseriti valori random in tutti i parametri usati.

Inoltre, il funzionamento del server OPC-UA è stato verificato con un applicativo di terze parti per controllare le sequenze in partenza dal server e quindi verificarne l'acquisizione da parte del client in run su SdM2.

Della definizione del server fa parte il suo ***endpoint*** rappresentato dall'indirizzo tcp del host che lo ospita, nonché dalla porta a cui accedere.

Altri parametri fondamentali sono i ***node space*** ed ***index*** dei parametri che il server trasmette. Prima della messa in funzione di SdM2 bisognerà indagare sul server reale per poter settare correttamente il funzionamento del client.

Un altro parametro importante da tenere in considerazione è il ***servicelevel***. Fornisce informazioni al client a riguardo della health del server e la sua abilità a fornire dati. L'algoritmo per la determinazione di questo valore è a carico del server.

4.3 TEST CON EUGENIO

Le informazioni che vengono inviate verso la piattaforma COMESTO devono passare attraverso EUGENIO.

Come già detto, i dati vengono incapsulati in un messaggio di tipo stringa codificato secondo il formato JSON (JavaScript Object Notation) e scambiati con EUGENIO tramite MQTT. I dettagli sulla struttura dei JSON e sullo schema adottato per la definizione di tutti i parametri che ne fanno parte, sono descritti nel documento [1].

Una prima verifica è stata condotta collegando EUGENIO alla scheda di rete di un PC connesso alla rete locale. Insieme a personale di EVOLVERE è stata provata la pagina di "inizializzazione" inserendo le credenziali generate appositamente per Spintel. Ciò ha consentito di verificare la sua raggiungibilità dall'esterno della rete locale. A valle di un esito positivo EUGENIO è stato connesso via cavo al router che genera la rete locale negli uffici della Spintel. È raggiungibile digitando http://sentinel:3000 nel browser di un PC in rete.

Come verifica propedeutica, è stato condotto un test apposito per verificare il corretto meccanismo di ***mDNS***. A tal proposito, dopo aver installato e configurato i servizi necessari su SdM2, sono state installate delle app apposite su un telefono cellulare (Service Browser, Bonjour Browser) per scoprire la presenza dei servizi che consentono di utilizzare il meccanismo di discovering. In tal modo è stato possibile verificare la presenza di servizi attinenti sia a ***Sentinel*** che al broker che gira su SdM2.

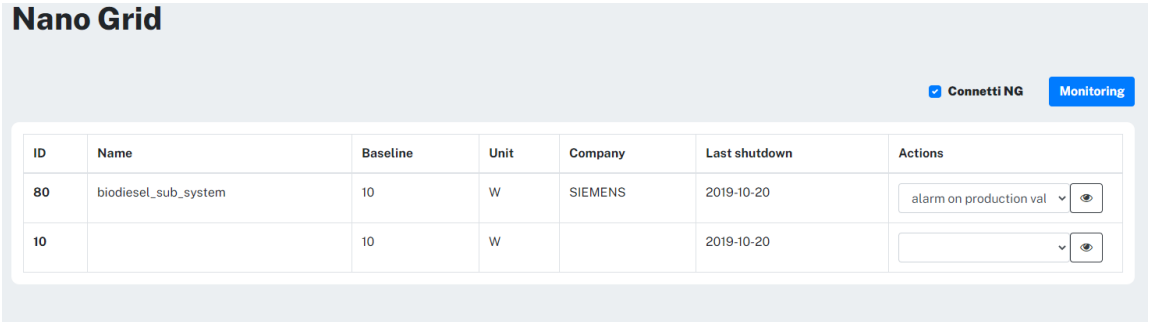
Con l'aiuto del personale di EVOLVERE si è potuto proseguire nel test per verificare la corretta interazione tra EUGENIO ed il broker installato su SdM2.

Si è iniziato col verificare le impostazioni sulle varie tab che EUGENIO mette a disposizione tramite un web browser. Inserite le credenziali per accedere al broker disponibile su SdM2 e spuntato il checkbox [NG] non si è riscontrato il meccanismo che ci si aspettava: EUGENIO pubblica la richiesta "get_subs" e attende la risposta;

quando questa arriva, valorizza i campi sulla form che riporta i dati riguardanti i sottosistemi comunicati da SdM2.

Una verifica locale, effettuata con un client javascript, invece dava un risultato positivo: SdM2 inviava la risposta corretta al client javascript. Quando EUGENIO rileva una connessione attiva, auto valorizza la checkbox "connetti ng". Nonostante le credenziali fossero giuste, riavviando EUGENIO, il checkbox "connetti ng" non rimaneva valorizzato. Da una verifica sui log di EUGENIO, veniva trovato un IP di un dispositivo con estensione <.local>. Ci si è accorti, però, che veniva rilevata una stampante presente negli uffici Spintel, che presenta la stessa estensione <.local>. Sistemato questo problema, ci si è accorti che ng.local non veniva individuato tra i servizi HTTP. Il problema è stato risolto editando il nome attribuito al servizio HTTP: compariva come “ngng.local” piuttosto di ng.local.

A valle di questi aggiustamenti è stato possibile verificare che SdM2 risponde correttamente alle richieste provenienti da EUGENIO. In particolare, l’operazione verificata è <get_subs>, richiesta pubblicata da EUGENIO ogni 10 secondi e che deve ritornare tutti i sistemi di produzione gestiti dalla NG (nella fattispecie i sistemi di accumulo non convenzionale collegati con SdM2). Al momento del test erano inseriti dei valori indicativi, ma la tabella resa da EUGENIO è stata valorizzata correttamente, come si può verificare dalla immagine seguente.



Naturalmente <get_subs> non è l’unica operation a cui il broker che risiede su SdM2 deve rispondere. Pertanto è stato creato uno java script ad-hoc per poter verificare tutte le altre operations previste in [1]: <check>, <get_state>, <set>, <reboot>, <shutdown>, <set_profile>.

Lanciando una operation alla volta è stato verificato che la risposta del broker fosse quella attesa.

5 CONCLUSIONI

L’attività svolta ha portato alla definizione, per come previsto in progetto, ai prototipi dei sistemi di acquisizione SdM1 e SdM2 da integrare nel sito pilota oggetto di attività dell’OR7.

6 BIBLIOGRAFIA

[1] AR. 4.1.3 - P-NG: Protocollo di comunicazione M2M per la Nano Grid