Protocollo LAXE

Presentazione

Protocollo LAXE

“Sistema elettronico modulare safety critical e fault tollerant per la verifica della corretta esecuzione di operazioni critiche eseguite dal sistema stesso. “

Brevetto numero : 102018000007412

Autore: Ing. Giovanni Galimberti.

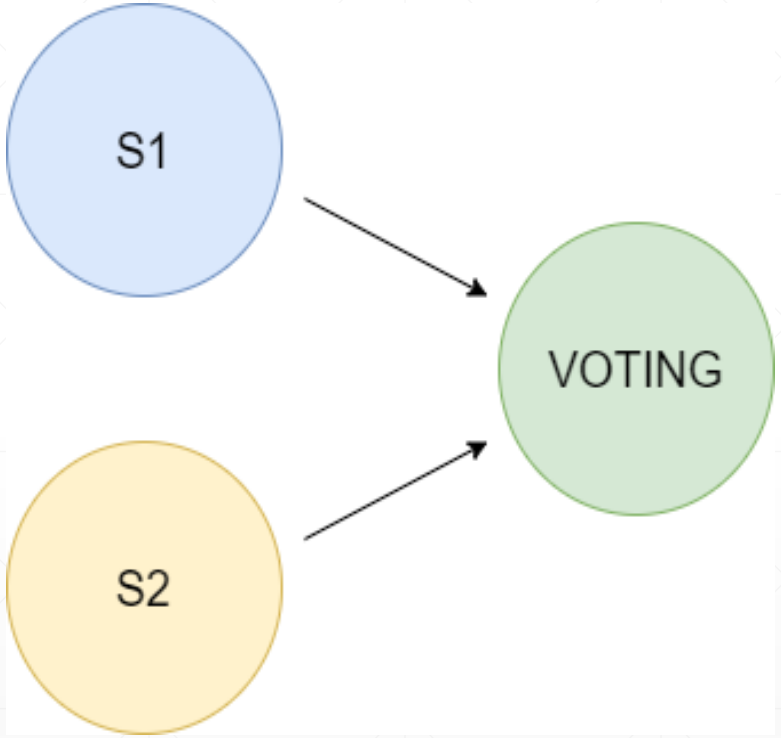
Introduzione

- Al fine di assicurare un determinato grado di affidabilità delle operazioni un sistema deve essere progettato in maniera tale che ogni guasto abbia una influenza controllata sulla affidabilità e sulla sicurezza di funzionamento del sistema stesso.
- Valutare l'affidabilità di un'architettura software è molto difficile: i sistemi oggi in uso hanno problemi di "pesantezza", complicazione e costi elevati, pertanto sono di attuazione onerosa in quanto debbono essere basati su strumenti singolarmente certificati.

Stato della tecnica

- Una normale realizzazione prevede un sistema duplicato, alimentato da situazioni di ingresso diverse, ed una logica a valle con il compito di confronto e verifica della correttezza delle elaborazioni a monte (detto Voting).

- Le limitazioni di questa architettura risiedono nella presenza di una parte singola, non controllata, (hardware o software) il cui malfunzionamento implica un malfunzionamento dell'intero sistema.



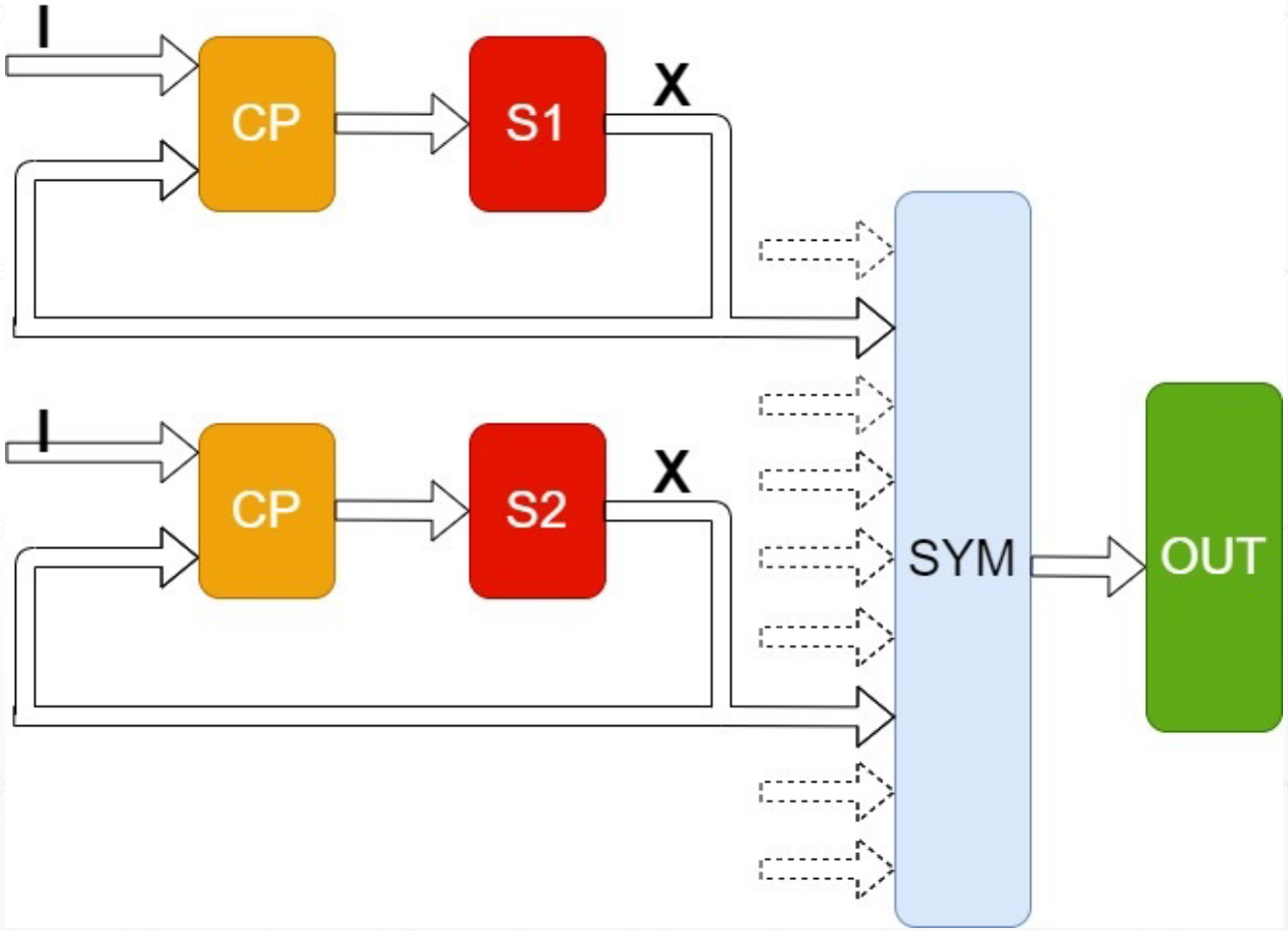
Il Protocollo LAXE

- Scopo del presente protocollo è quello di:
  - Realizzare un sistema elettronico modulare per la verifica della corretta esecuzione di operazioni eseguite dal sistema stesso.
  - Realizzare un metodologia di Voting distribuita a livello HW.
- Si tratta di un innovativo “software concept”. Il sistema raggiunge l’affidabilità e la sicurezza di funzionamento richiesta utilizzando un’architettura modulare in grado di fornire evidenza che il sistema funzioni correttamente tramite il confronto continuo dei risultati delle elaborazioni eseguite su diverse e differenti unità.

Aree dati di utiliizzo

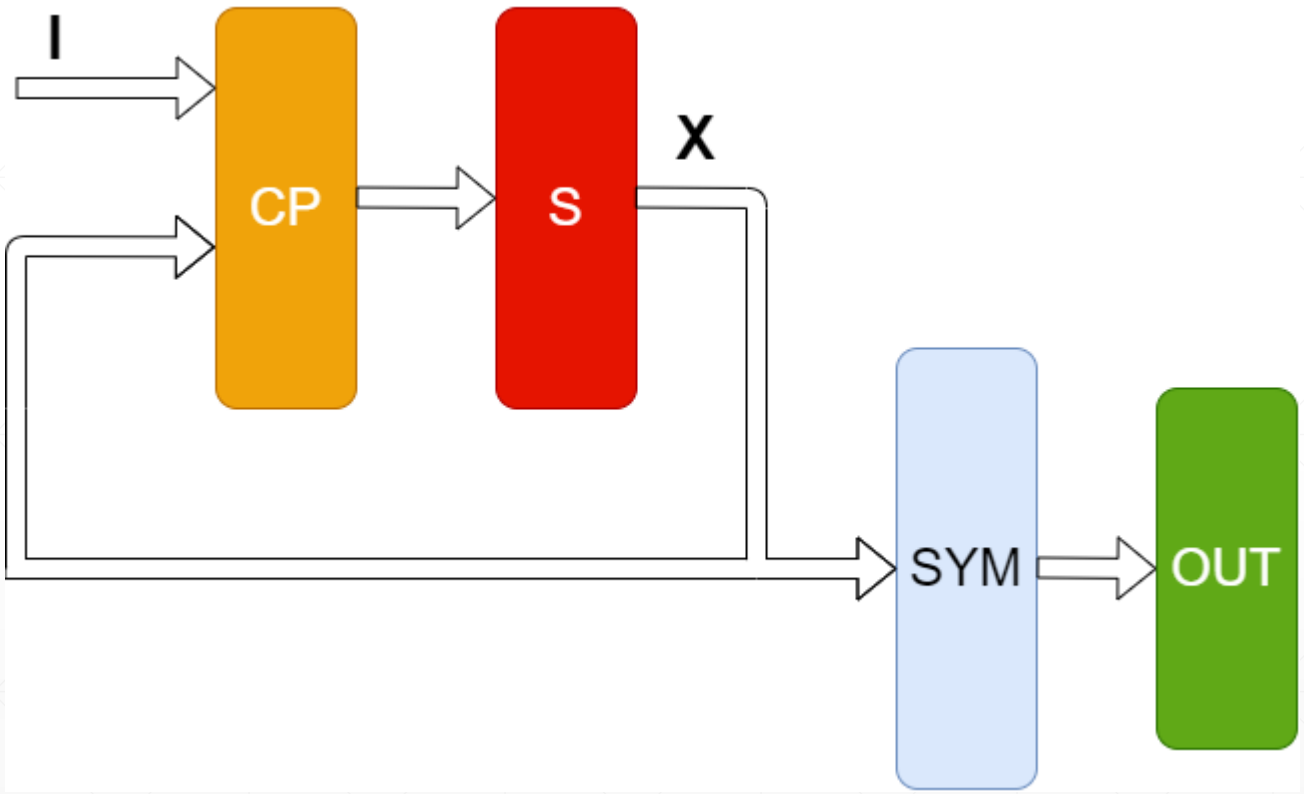
- Al fine di schematizzare il sistema si ipotizza che il sistema sia riconducibile ad una macchina a stati. Ipotizziamo di mettere tutte le variabili statiche in una struttura, chiamata in seguito processo , composta essa stessa da tre ulteriori strutture: ingresso **I** , stato **Sx**, uscite **OUT**.

- Le uscite dipenderanno da tutti gli stati del sistema (**S1,S2**) come illustra la figura a fianco



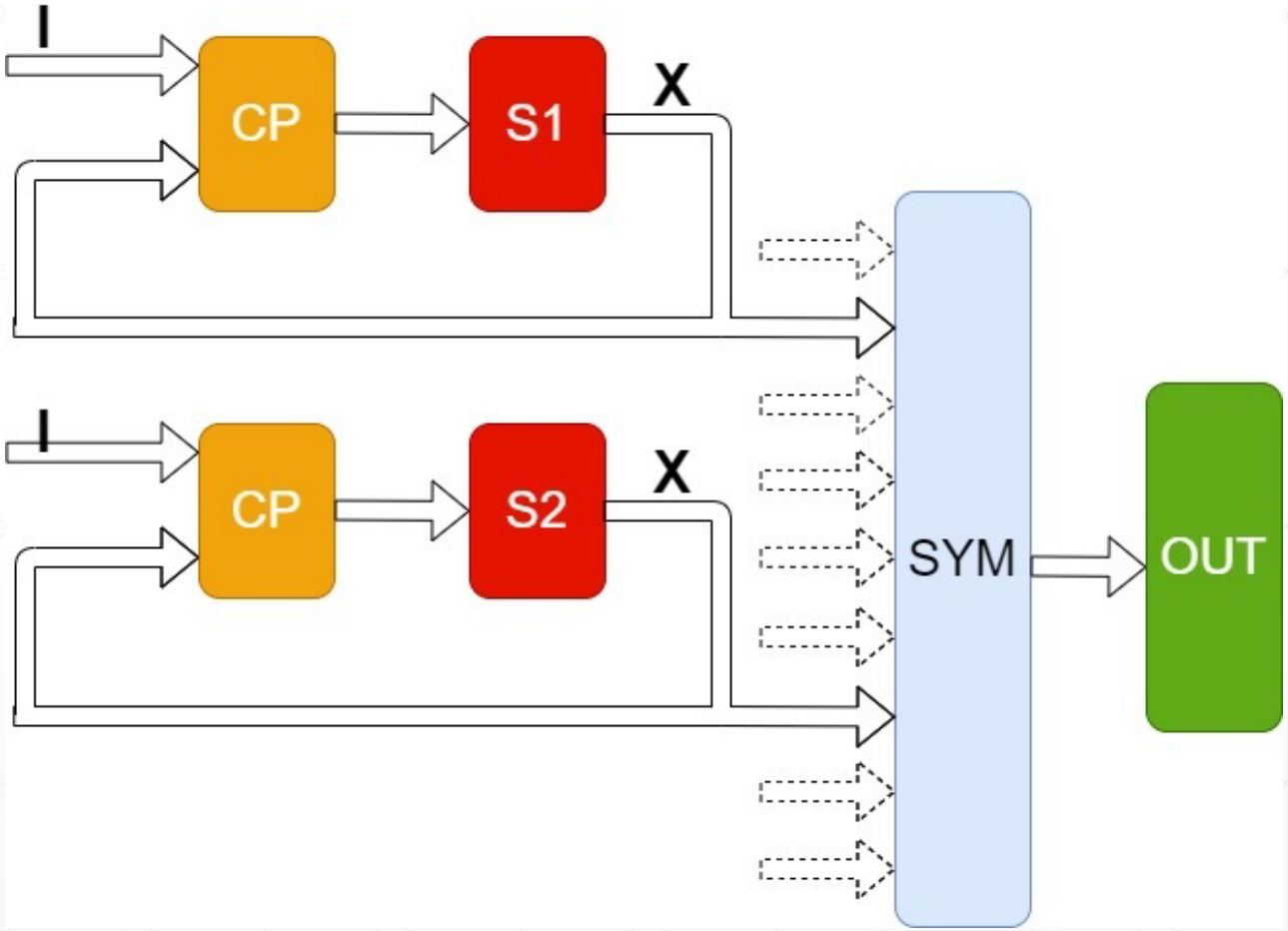
Macchina a stati (creazione stato futuro)

- Ogni singolo processore campiona gli ingressi **I** e tramite la funzione **CP** crea lo stato futuro **S**: X(t+1) =f[(X(t), I(t)].
- La macchina produce quindi dei vettori di stato che rappresentano tutta la sua storia.



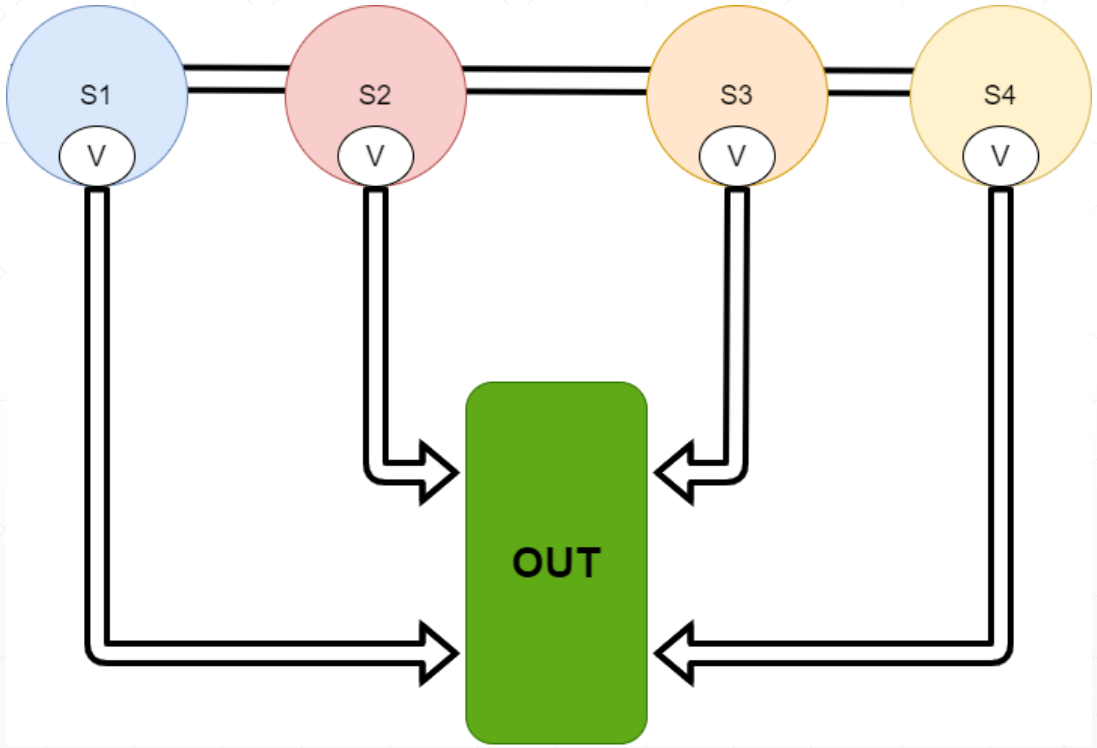
Macchine a stati (creazione uscite simmetriche)

- Utilizzando N sistemi indipendenti, identici a livello di Sw ma potenzialmente diversi a livello di Hw, ognuno con il proprio vettore di stato, si pone il problema di sincronizzarne i risultati che tenderanno a divergere al variare temporale della registrazione ingressi.
- Risulta pertanto necessario avere un algoritmo SYM per produrre delle uscite comuni e condivise.



Sincronizzazione

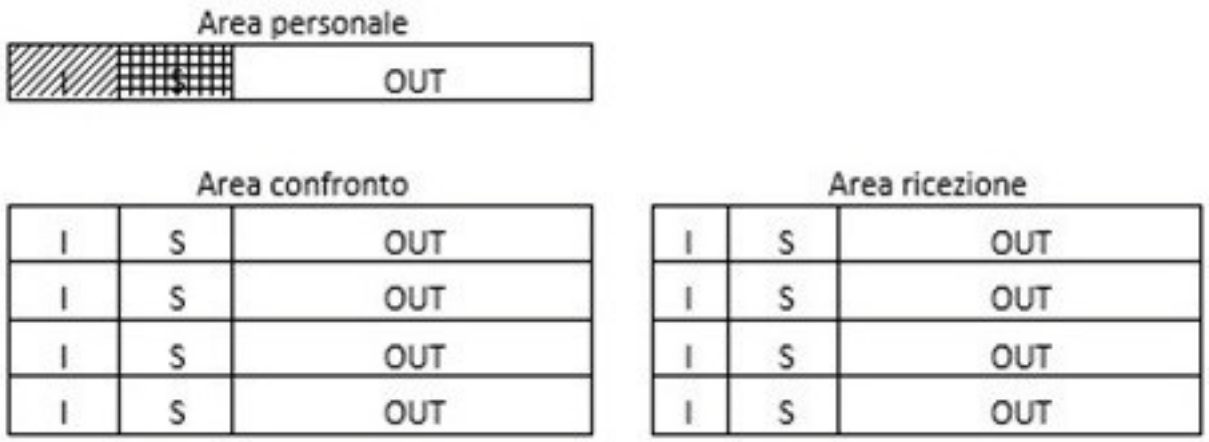
- Per ottenere tale sincronizzazione ogni sistema lavora su tre fasi.
- Inizialmente ognuno fotografa ed elabora la sua realtà, successivamente vi è uno scambio degli ingressi, degli stati e degli out tra i sistemi e un ripetersi delle operazioni fino a quando ogni sistema non ha ripetuto le stesse elaborazioni di tutti gli altri. Si ha quindi certezza che le conclusioni raggiunte siano controllate.
- Inoltre viene applicato per ogni sistema un algoritmo simmetrico che determina il risultato ufficiale delle uscite.



Fase 1 - Lettura ingressi, elaborazione e trasmissione

- In questa fase accadono le seguenti azioni:

  1. Lettura ingressi (caricamento area ‘I’);
  2. Calcolo stato futuro in funzione di stato presente ed ingresso (calcolo area ‘S’);
  3. Trasmissione dati I/S/OUT agli altri processori (l’area personale) (trasmissione area ‘I’ + ‘S’ + ‘OUT’).



Fase 1 - Lettura ingressi, elaborazione e trasmissione

- Ognuno dei sottosistemi legge dall'esterno un dato di ingresso **I**, lo elabora tramite l'algoritmo **CP** (CriticalProcess) e calcola lo stato futuro **S**.
- Nell'algoritmo CP sono implementate le specifiche del sistema (fase evoluzione)
- Ogni unità esegue le medesime operazioni utilizzando i propri ingressi e conseguentemente si possono generare tra i vari processori delle sequenze di stato diverse.
- Il valore dello stato **S** all'istante **n+1** è funzione **f** dello stato **S** all'istante **n**, e dell'ingresso acquisito in quell'istante secondo la classica relazione: $X(t+1) = f[(x(t), I]$

Fase 1 - Lettura ingressi, elaborazione e trasmissione

- L'ingresso **I** è teoricamente uguale per tutti gli **N** processori. Di fatto nella realtà operativa si possono verificare piccole diversità dovute a specifiche situazioni, quali campionamento in istanti diversi tra loro, o variazione di parametri locali del sistema tipo frequenza del quarzo o precisione di eventuali ingressi analogici.

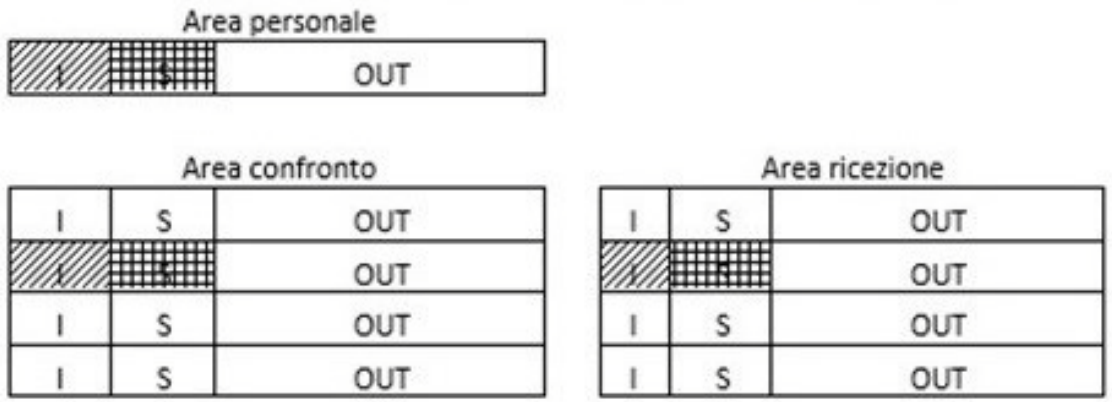
- Queste piccole diversità possono creare dei percorsi divergenti fra gli stati dei vari sottosistemi.

- Risulta necessaria una metodologia per ricavare uscite identiche per tutti i sottosistemi.

Fase 2 – Ricezione da altri processori, controllo elaborazione altrui.

- Questa fase viene attivata ad ogni singola ricezione del pacchetto dati I/S/OUT trasmesso.
- In questa fase accadono le seguenti azioni:

1. Ricezione dati in area ricezione.
2. Copia solo ingressi da ricezione a area confronto (caricamento area ‘I’)
3. Calcolo stato futuro in funzione di stato presente ed ingresso su area confronto (calcolo area ‘S’)
4. Confronto dati fra ricezione e confronto (Confronto aree ‘S’)



Fase 2 – Ricezione da altri processori, elaborazione, controllo

- In questo step vengono copiati gli ingressi del sistema, viene eseguito l'algoritmo CP ottenendo in S lo stato futuro e viene verificato che tutte le elaborazioni dei sistemi coincidano, questo controllo verifica la corretta evoluzione dello stato.
- Alla ricezione del messaggio dal processore A vengono eseguiti da tutti i processori i medesimi calcoli che il processore A aveva eseguito localmente nella sua area privata, utilizzando i medesimi ingressi.
- Questa azione garantisce che l'evoluzione dello stato S sia la stessa (bit a bit) dallo start del sistema.
- Si sottolinea che i processori possono essere di tipo diverso con sistemi operativi diversi ed anche con un codice oggetto diverso, ottenuto però dal medesimo sorgente.
- Un successivo controllo (Fase 3) verificherà l'uguaglianza delle uscite simmetriche.

Fase 3 – Ricezioni di tutti, controllo uscite simmetriche, calcolo nuove uscite simmetriche.

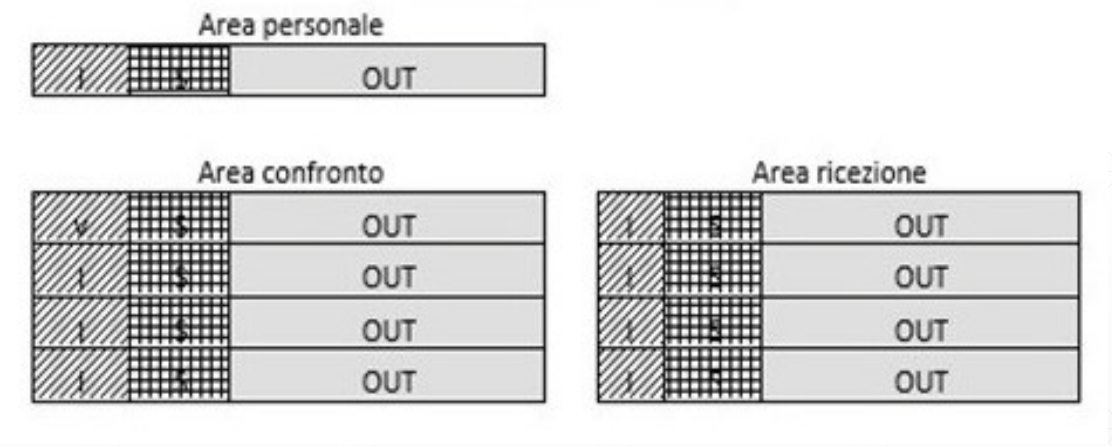
- Questa fase viene attivata una volta ricevuti, elaborati e confrontati tutti i pacchetti dati I/S/OUT trasmessi.

- In questa fase accadono le seguenti azioni:

1. Controllo uscite simmetriche (Uguaglianza di tutte le aree 'OUT'); (tutti i campi OUT di area confronto devono essere identici bit a bit al campo OUT personale).

2. Calcolo delle nuove uscite simmetriche ovvero creazione del nuovo campo OUT nell'area personale che sarà successivamente trasmesso alla nuova fase 1 insieme alle attivazione uscite.

3. Controllo che tutti i processori abbiano ricevuto i dati da tutti gli altri per iniziare nuovo ciclo.



Fase 3 – Ricezioni di tutti, controllo uscite simmetriche, calcolo nuove uscite simmetriche.

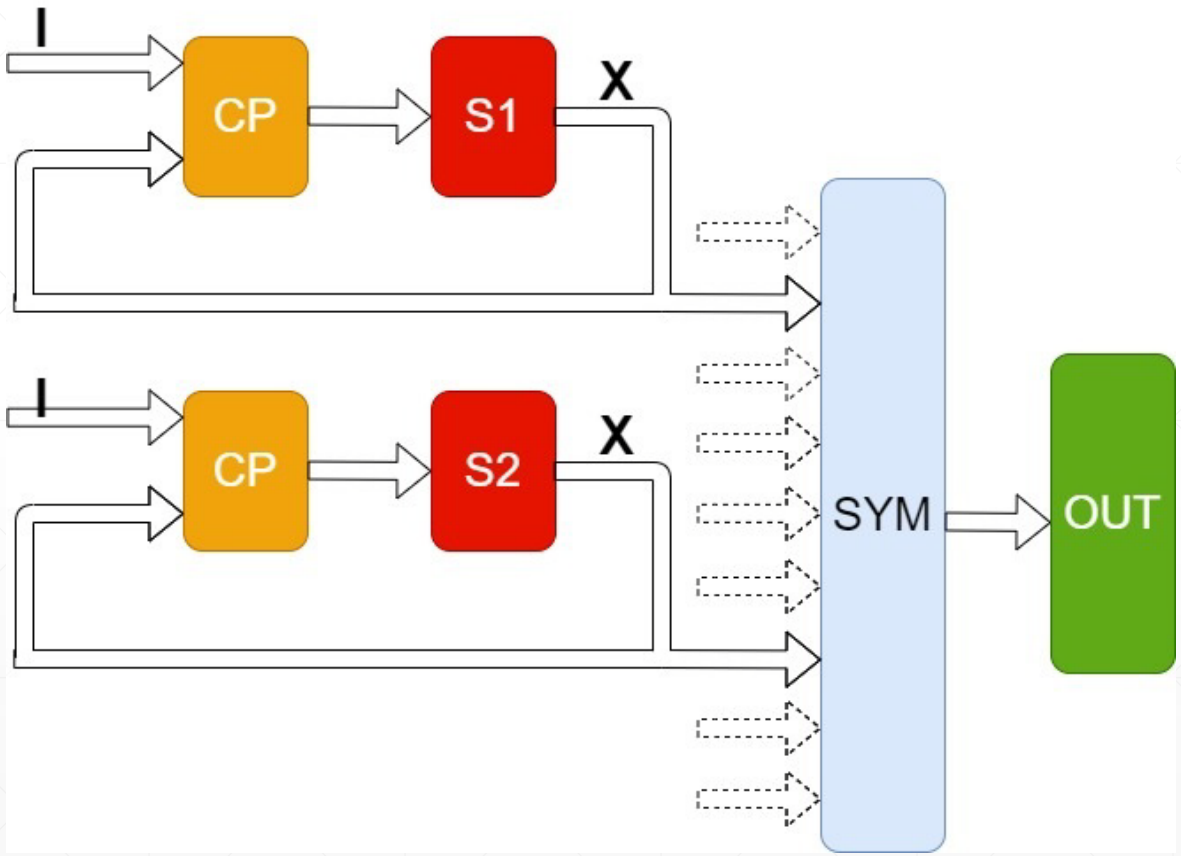
- In questa fase le aree confronto sono già state caricate (ingressi) e verificate (stato e uscite) come da fase 2. Vengono utilizzate le aree ricezione e confronto, viene creata l'area **OUT** personale e viene applicato da ogni sistema l'algoritmo di voto simmetrico. Il risultato del voto viene quindi confrontato con il risultato ottenuto dagli altri sistemi.

- Il campo **OUT** verrà trasmesso nel punto **1** di tale sequenza.

- La prima volta, in effetti, si confrontano esclusivamente i valori di inizializzazione del campo **OUT**.

Fase 3 – Ricezioni di tutti, controllo uscite simmetriche, calcolo nuove uscite simmetriche.

- L'algoritmo SYM deve dare le medesime uscite per ogni tipo di permutazione degli stati dei processori.
- Questo risulta necessario per avere su tutti i sottosistemi le medesime uscite OUT che in questo passo sono verificate bit a bit.
- Nella terza Fase si ha quindi un ulteriore controllo di sincronismo tra i processori.



Note

- In definitiva si può dire che la funzione **f** sopra definita è realizzata tramite un software che simula una rete combinatoria alle cui uscite presentano registri in logica sequenziale con memoria e retroazione all'ingresso. Il risultato delle elaborazioni viene ricalcolato e confrontato da ogni processore eseguendo un duplice controllo.

- E' possibile scegliere tra diverse alternative per risolvere lo scambio dati tra i processori. Si è scelto di inviare un solo messaggio per ogni coppia di sottosistemi, contenente tutti i campi I/S/OUT.

- Questa scelta risulta vantaggiosa per sistemi semplici, infatti in questo caso il campo I/S/OUT è circa 130 bytes per cui conviene trasferirlo in un solo messaggio. Come controindicazione si ha però il fatto che il confronto del campo OUT viene fatto nel passo successivo.

Architettura Sw

- L’architettura Software realizza un gestore generico di algoritmi (SequenceController) atto a garantire la corretta esecuzione di un algoritmo (CriticalProcess) tramite esecuzione parallela su più sottosistemi hardware/software, sulla base delle seguenti ipotesi:

1. Architettura Hardware basata su più processori di tipologia diversa.
2. Lettura degli ingressi del sistema da parte di ciascun processore.
3. Esecuzione dell’algoritmo (CriticalProcess) su tutti i processori in parallelo, con ingressi diversi.

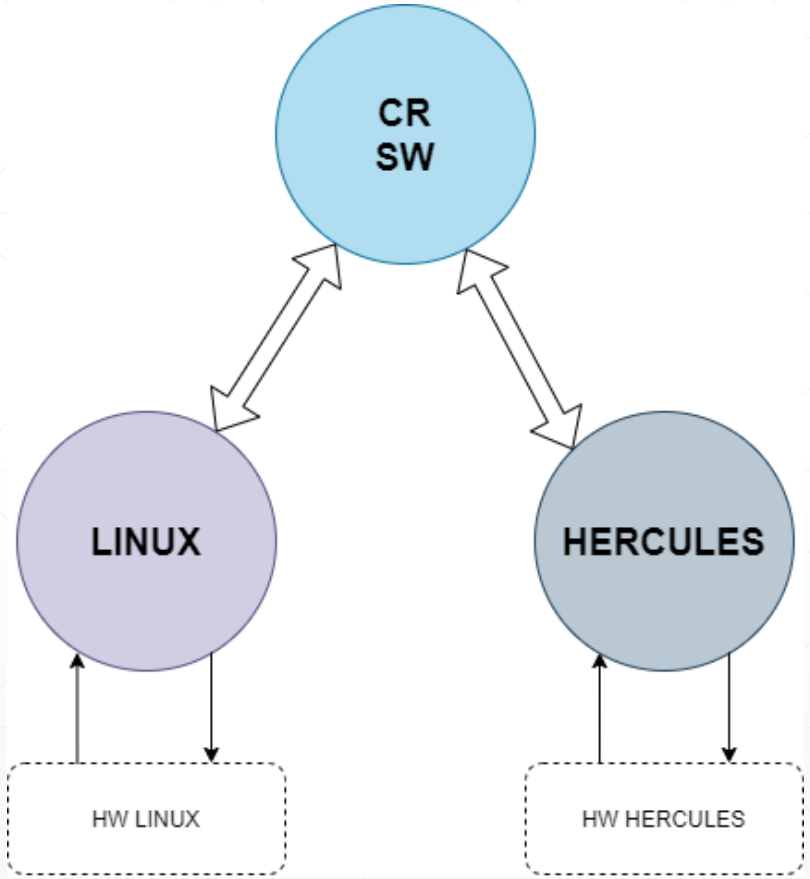
Architettura Sw

4. Vi è uno scambio dati fra i vari processori.
5. Ri-esecuzione da parte di ciascun processore con i dati di ingresso ricevuti da tutti gli altri, utilizzando il medesimo algoritmo o la medesima funzione.
6. Confronto della corretta elaborazione.
7. Calcolo di risultati comuni tramite algoritmo simmetrico (SymmPostElab).

Architettura Sw - Compilazione

- Questa sequenza permette l'elaborazione di un algoritmo in maniera virtuale perché viene eseguito e confrontato su processori diversi, con l'eseguibile ottenuto tramite percorsi di compilazione diversi, per cui i risultati del generico algoritmo possono ritenersi corretti anche se non viene certificata la correttezza del compilatore o del sistema operativo o del protocollo di comunicazione.

- Nell'immagine d'esempio la parte '**CR**' viene compilata con due compilatori diversi e gira su due sistemi operativi diversi; inoltre le funzioni di accesso all'HW sono diverse una per 'LINUX' ed una per 'HERCULES'.



Case Study: USMM

- L’apparato USMM (Unità per la Sicurezza dei Movimenti di Manovra) è un prodotto concepito per rispondere ai requisiti imposti dal decreto ANSF n. 1/2015 relativamente a locomotive di manovra di tipo B e treni storici.

Funzioni di Sicurezza

L'USMM, una volta installato sulle locomotive in sostituzione delle analoghe apparecchiature elettromeccaniche pre-esistenti, realizza le seguenti funzioni di sicurezza:

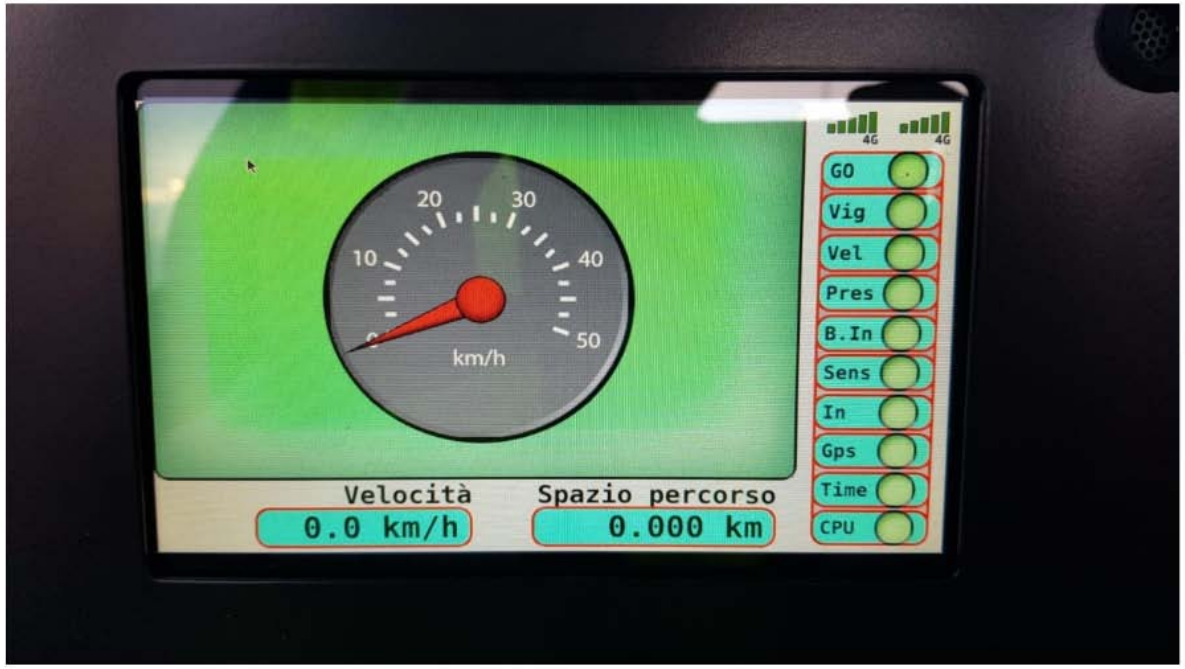
- Controllo della velocità massima a 30 km/h e, al superamento della stessa, taglio trazione e frenatura di emergenza secondo norme ANSF.
- Controllo della presenza dell'agente di condotta alla perdita della condizione di rotabile fermo.
- Registrazione degli eventi e dei parametri relativi alla condotta e ai movimenti di manovra sia in locale che su Cloud.
- Controllo della vigilanza dell'agente di condotta (uomo morto).
- Controllo della pressione della condotta freno all'avvio del rotabile e durante la marcia.
- Controllo posizione locomotore tramite gps.

Composizione (1)

- una centralina contenente 2 schede identiche, ciascuna contenente 2 processori
- La centralina è un singolo apparato composto da due schede elettroniche gemelle inscatolate in un case metallico con connessioni atte al mercato ferroviario

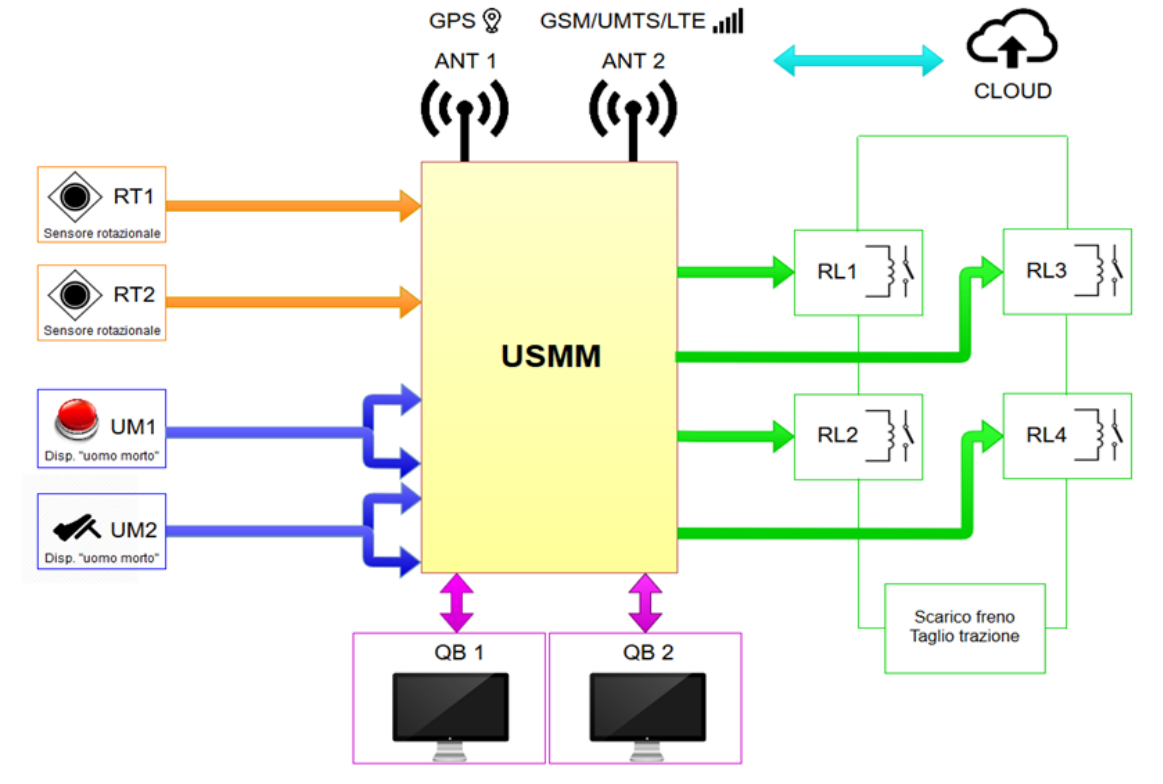
Composizione (2)

- Due display per la visualizzazione dati di condotta (Qbe)
- Antenna Gps+Lte
- Meccanica
- Cloud



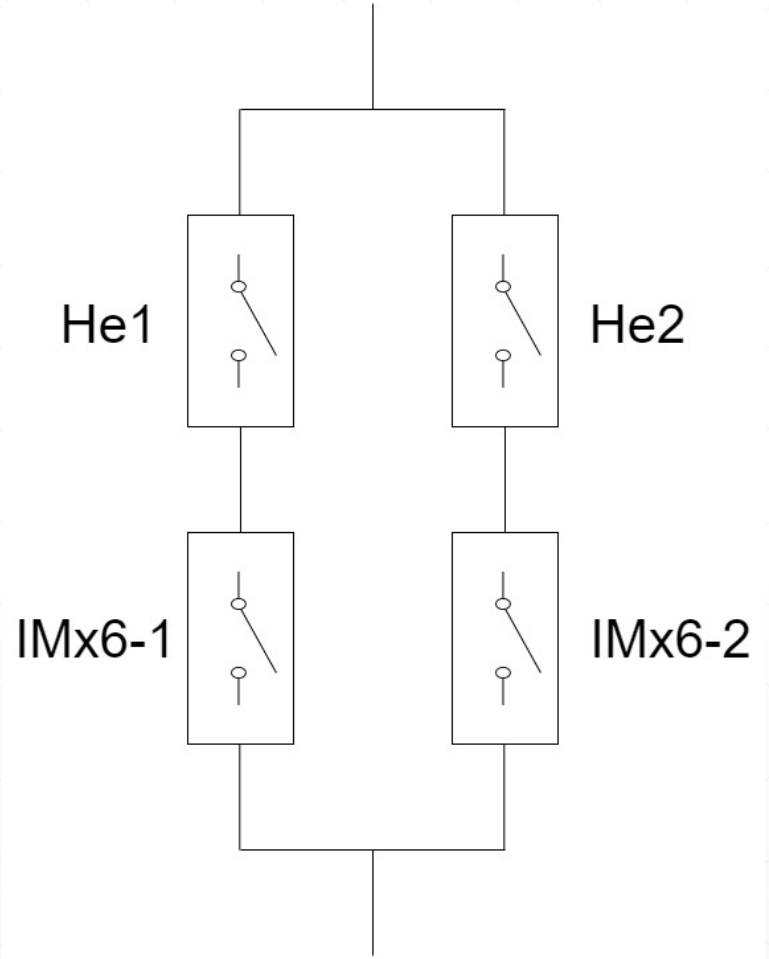
Composizione (3)

- Relè di controllo frenata e trazione
- Pulsanti ''Uomo Morto''
- Sensori/Encoder



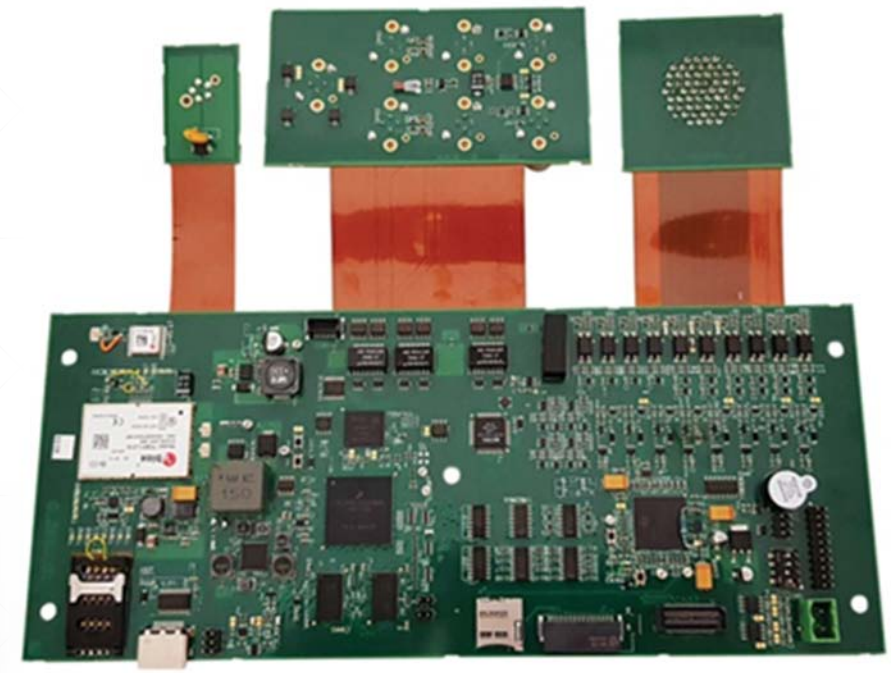
Composizione (4)

- Relè SIL normalmente aperti, controllano la pressione alla condotta dei freni e alla trazione, facendo intervenire il sistema di arresto. Per permettere il movimento del treno, i relè devono dunque essere comandati (chiudendo il contatto) dalla scheda.
- La sicurezza del sistema è perciò intrinseca: qualsiasi malfunzionamento della scheda che comporti il mancato pilotaggio dei relè provoca l'apertura del contatto e l'intervento della frenatura.



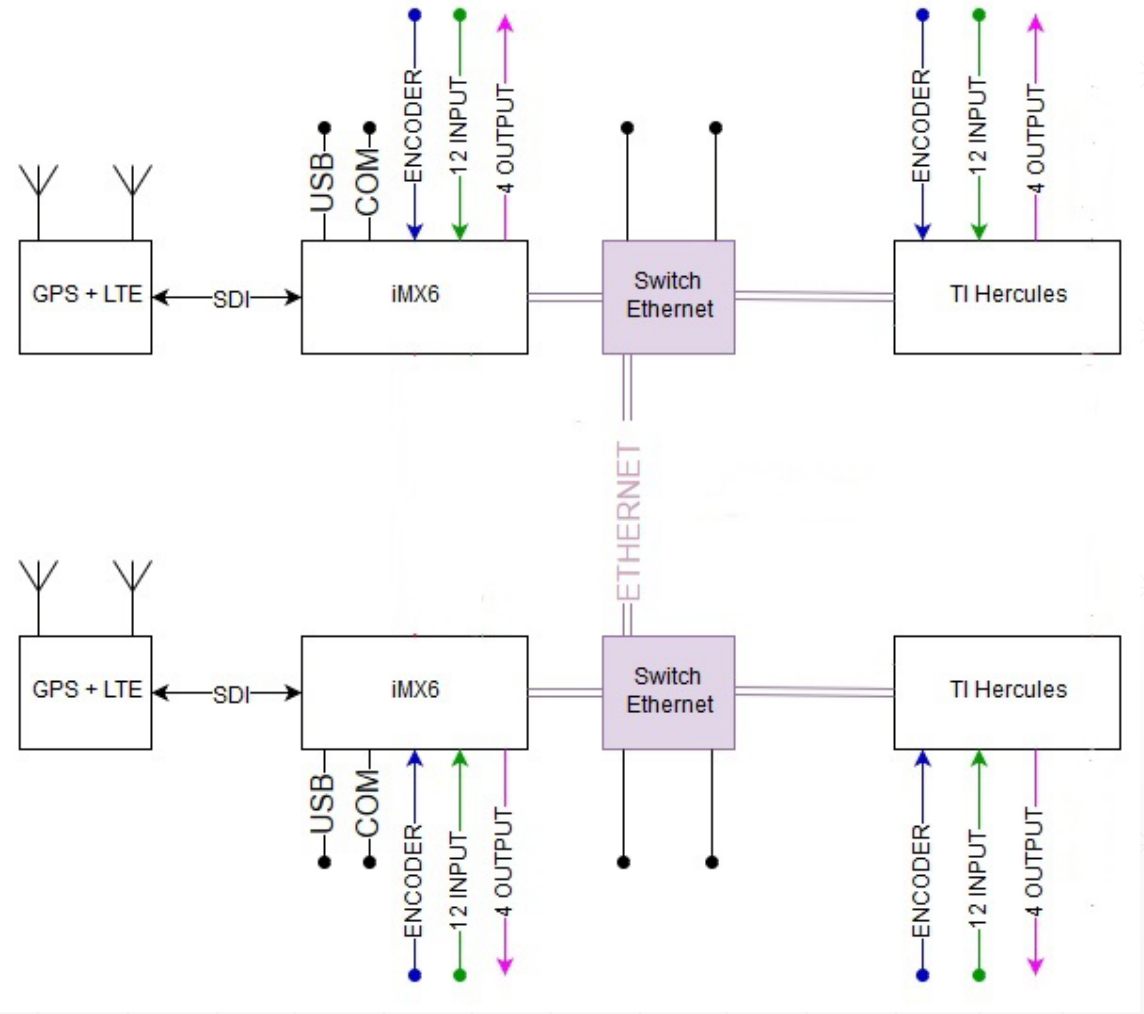
Le schede (1)

- Ciascuna scheda è dotata di un microcontrollore della famiglia TI Hercules (RM48L852), Cortex R4F e di un microprocessore della famiglia i.MX6 (Cortex-A9 Dual Core), interconnessi tramite uno switch ethernet on board



Le schede (2)

- L'USMM riceve in ingresso segnali digitali per la funzionalità di uomo morto: essi sono acquisiti e processati sia dal microcontrollore, sia dalla CPU iMX6
- Allo stesso modo, anche i 4 segnali di uscita (i comandi per la frenatura di emergenza) sono duplicati e gestiti da entrambe le CPU: è sufficiente che solo uno dei due processori individui una situazione di pericolo (ad es. il superamento della velocità massima consentita) per arrestare il treno



Processori

- Il firmware presente sul microcontrollore Hercules è progettato per permettere le migliori prestazioni e interfacciarsi con l'hardware dedicato tramite periferiche appositamente realizzate. La peculiarità di questo microcontrollore consiste nell'essere dotato di architettura "**lockstep**": due core, identici ma indipendenti, eseguono le stesse istruzioni; in tal modo, eventuali errori nell'esecuzione del codice vengono individuati e gestiti.

- La Cpu IMx6, basata su architettura Linux fornisce un sistema di calcolo alternativo ed oltre a controllare le stesse funzionalità dell'Hercules gestisce anche la connessione Gps e Cloud

Display (1)

Sui display vengono visualizzate le seguenti informazioni:

- Velocità
- Allarmi
- Diagnostica di sistema

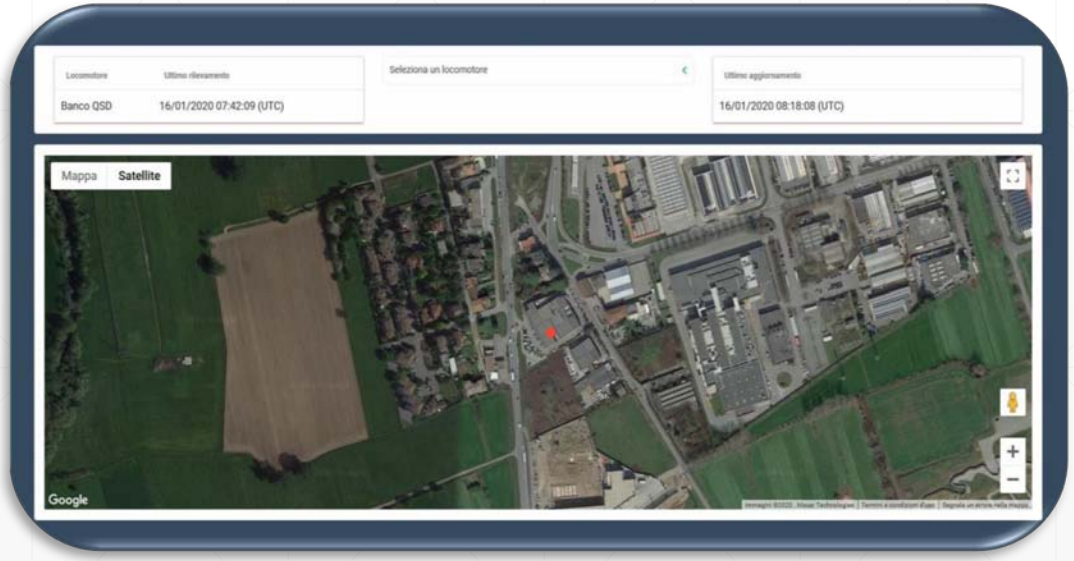
- L'USMM può connettersi tramite l'interfaccia Ethernet ad un display touch screen (Qbe) per poter svolgere oltre alle normali funzionalità di informazione, come la visualizzazione della velocità, anche altre funzionalità di interoperabilità con il sistema.
- Ogni scheda è connessa ad un proprio display. Le due schede sono collegate tramite ethernet per lo scambio dei dati.

Antenna

- Le schede possono interfacciarsi con antenne poste esternamente alla motrice. Tali connessioni sono realizzate utilizzando cavi certificati per l'utilizzo ferroviario e permettono la ricezione dei segnali Gps ed Lte da parte di ciascuna scheda, rilevando posizione e velocità in modo alternativo.

Cloud

- Il sistema grazie alle connessioni Lte è in grado di eseguire un backup su cloud del proprio registro eventi, l'identificazione in tempo reale della posizione della locomotiva e visualizzazione su mappa.



Meccanica

Tramite la meccanica il sistema USMM è in grado di interfacciarsi con il treno e determinarne velocità e spostamento.

Q.S.D. Sistemi offre due opzioni standard:

- Ruota fonica e sensori
- Encoder

La scelta della meccanica e la sua successiva realizzazione è a discrezione di Qsd Sistemi, avviene in seguito all'ispezione della motrice e può variare per ogni installazione.

Grazie per l’attenzione!