Supervisori

Febbraio 2026

Contents

1 Acquisizione segnali Board 2

1.1 HC-SR04 - Sensore ad ultrasuoni

La Board 2 è equipaggiata con tre sensori ad ultrasuoni **HC-SR04** per il rilevamento di ostacoli. Questi sono disposti a 45° l'uno dall'altro, come mostrato in Figura 1.

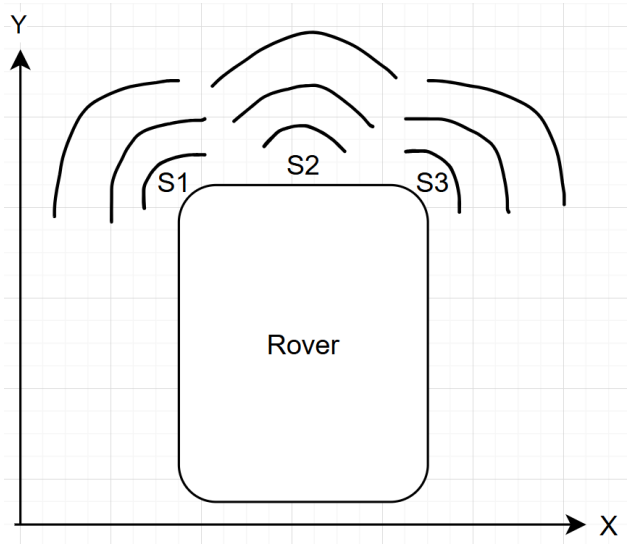


Figure 1: Disposizione dei sensori sulla Board 2.

Ogni sensore emette onde sonore ad alta frequenza e produce segnali di tipo onda quadra la cui durata è proporzionale all'ostacolo rilevato.

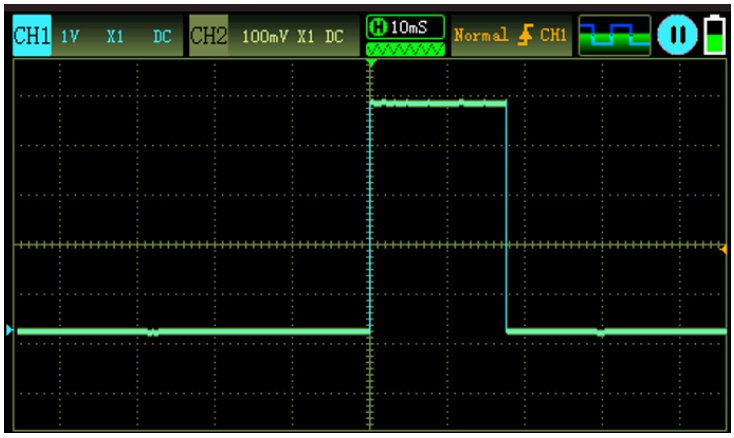


Figure 2: Segnale generato dal sensore HC-SR04 in presenza di un ostacolo a 3 metri di distanza.

La board2, rilevando i fronti di salita e discesa, può misurare l'intervallo tra i due fronti e utilizzare questa informazione per calcolare la distanza dall'ostacolo e prendere decisioni appropriate per evitare collisioni.

Utilizzo DMA per la lettura dei segnali

Per ottimizzare la lettura dei segnali dai sensori ad ultrasuoni, la Board 2 utilizza il Direct Memory Access (DMA). Il DMA consente di trasferire i dati direttamente tra la periferica (i sensori ad ultrasuoni) e la memoria, senza l'intervento della CPU. Il timer utilizzato è il $Timer1$, con i canali 1, 2 e 3 configurati in modalità *input capture* per catturare i fronti di salita e discesa generati dai tre sensori.

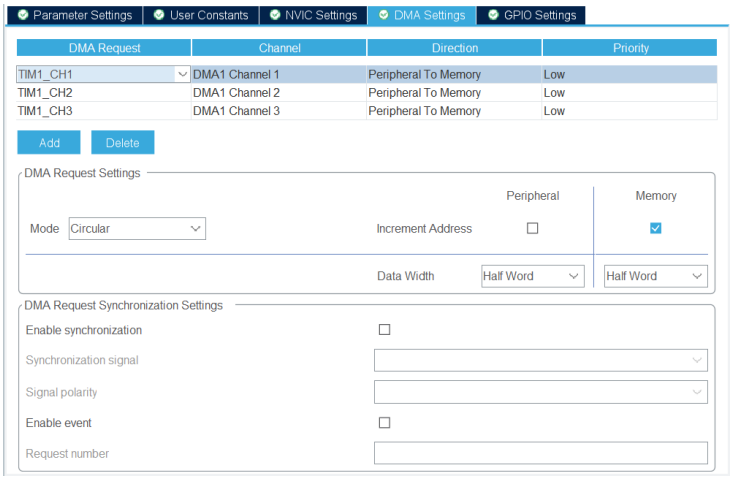


Figure 3: Configurazione del DMA per la lettura dei segnali dai sensori.

Ogni canale del DMA è configurato in modalità interrupt, permettendo, alla fine della rilevazione dei due fronti (salita e discesa), di eseguire una *Callback* che imposta dei flag a 1. Questo flag indica che i fronti sono stati rilevati e che la distanza dall'ostacolo può essere calcolata. In totale vengono eseguite solo 3 callback, attivate solo quando uno specifico canale DMA ha terminato la lettura di entrambi i fronti. La Figura 4 mostra un esempio di callback eseguita al termine della rilevazione dei fronti.

```c
void HAL_TIM_IC_CaptureCallback(TIM_HandleTypeDef *htim)
{
    BaseType_t xHigherPriorityTaskWoken = pdFALSE;

    if(htim->Instance == TIM1){
        switch(htim->Channel){
            case HAL_TIM_ACTIVE_CHANNEL_1:
                if(flag.sonar1_ok == 0){
                    flag.sonar1_ok = 1;
                    sonar_count ++;
                }
                break;
            case HAL_TIM_ACTIVE_CHANNEL_2:
                if(flag.sonar2_ok == 0){
                    flag.sonar2_ok = 1;
                    sonar_count ++;
                }
                break;
            case HAL_TIM_ACTIVE_CHANNEL_3:
                if(flag.sonar3_ok == 0){
                    flag.sonar3_ok = 1;
                    sonar_count ++;
                }
                break;
            default:
                    break;

        }
    }

    if (sonar_count >= 3) {
        // Notifica il task e richiedi uno switch immediato se necessario
        xTaskNotifyFromISR(sonarTaskHandle, 0, eNoAction, &xHigherPriorityTaskWoken);
        portYIELD_FROM_ISR(xHigherPriorityTaskWoken);
    }
}
```

Figure 4: Callback eseguita al termine della rilevazione dei fronti.

2 Supervisore Board 2

2.1 Panoramica generale

Il supervisore della Board 2 è implementato come un modulo Simulink denominato **SupervisorB2**, il cui schema a blocchi è illustrato in Figura 5.

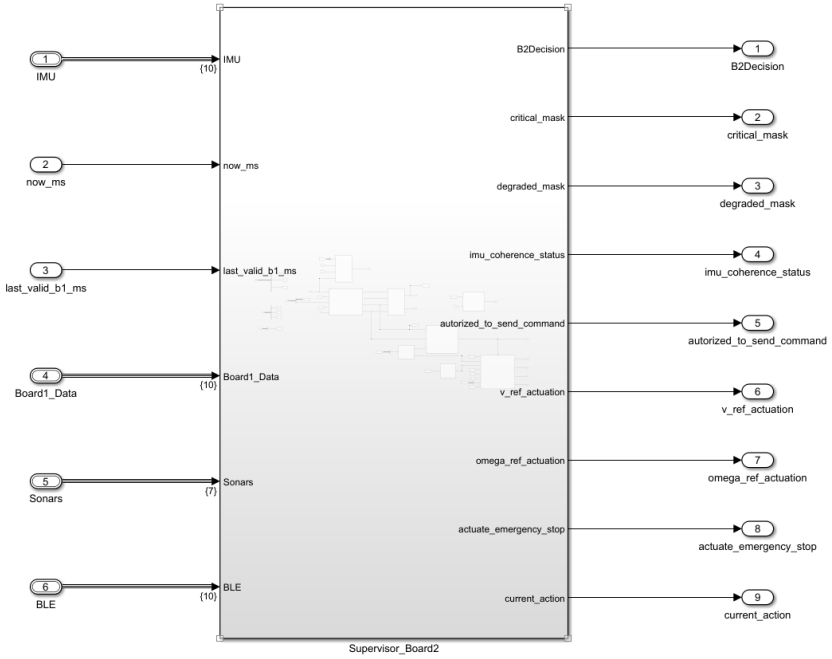


Figure 5: Schema a blocchi del modulo SupervisorB2.

Il suo compito è quello di decidere il riferimento di velocita (*v_ref*) e di direzione (*omega_ref*) del rover, in funzione dell'elaborazione dei dati di input. In particolare, esso è composto da 3 parti principali:

- **Gestione Faults**: si occupa di rilevare e gestire eventuali anomalie nei dati ricevuti dalla Board2, dagli encoder delle ruote, dai sensori di temperatura e batteria.
- **Decidere di far comandare la Board2**: decide se autorizzare o meno la Board2 a muovere il rover, in base alle condizioni di fault rilevate.
- **Aggregazione Fualt**: aggrega le anomalie rilevate nella parte di gestione faults e le codifica in due maschere di errore (critica e degradata).
- **Calcolo Riferimenti**: calcola i riferimenti di velocità lineare e angolare del rover in base ai comandi ricevuti dalla Board2 e alle condizioni di fault rilevate.

Nel seguito verranno descritti i segnali di input e output del supervisore, successivamente verranno descritte le tre parti principali del supervisore descritte sopra.

2.2 Input

I segnali in input che riceve sono:

2.3 Output

I segnali in output che fornisce sono:

2.4 Rilevamento ostacoli

Come da specifiche, il comportamento del rover in presenza di ostacoli, deve essere regolato sulla base delle condizioni in cui può trovarsi:

1. *Stato non degradato*
    - **Distanza dell'ostacolo $\leq$ 70 cm:** il rover deve fermarsi immediatamente per evitare collisioni.
    - **Ostacolo a distanza $>$ 100 cm in movimento tra due sonar:** il rover deve determinare la direzione dell'ostacolo e deve deviare il percorso di conseguenza in direzione del sonar che per prima ha rilveato l'ostacolo.
2. *Stato degradato*
    - **Distanza dell'ostacolo $\leq$ 300 cm:** il rover deve fermarsi immediatamente per evitare collisioni.

In seguito verranno mostrati i chart realizzati per la gestione delle due casistiche.

2.4.1 Gestione ostacoli con sistema in stato *non degradato*

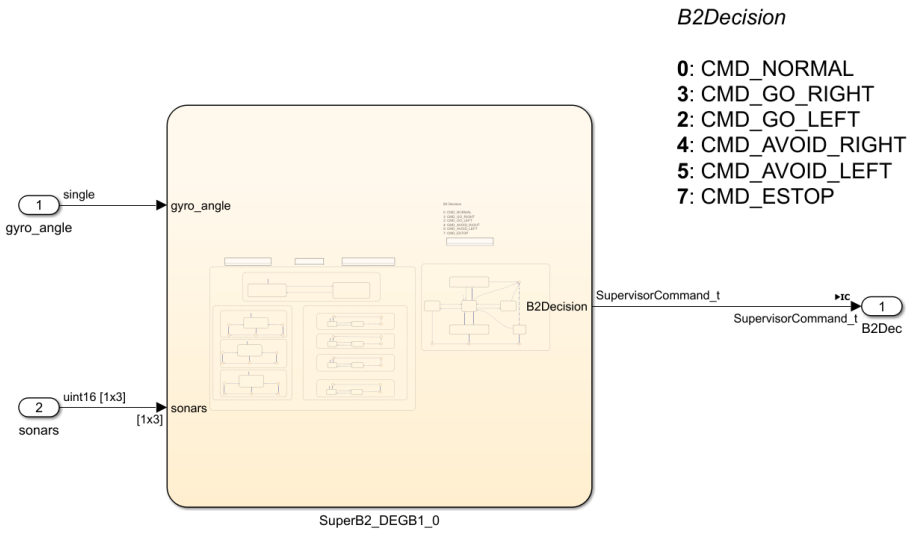


Figure 6: Logica di gestione degli ostacoli in stato non degradato.

Il chart per la gestione degli ostacoli in stato non degradato è mostrato in Figura 7.

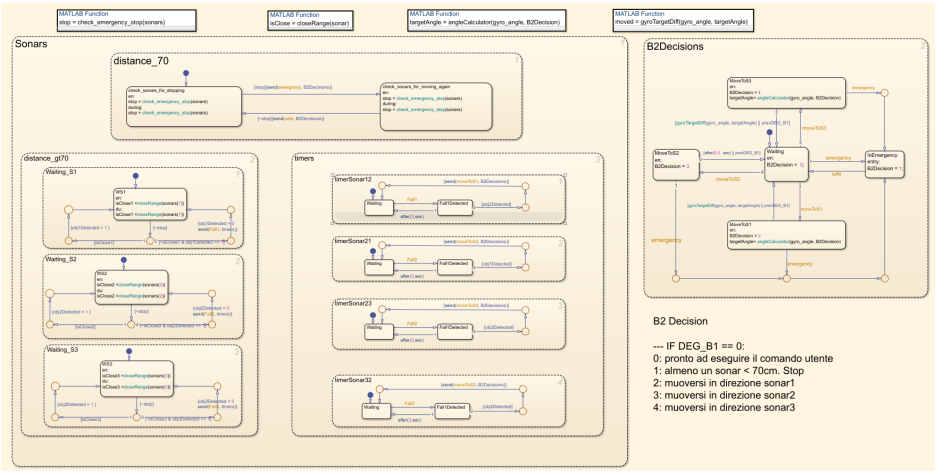
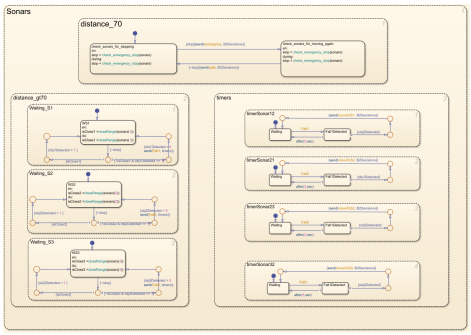


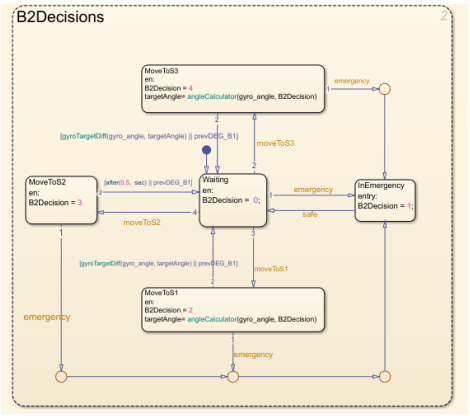
Figure 7: Chart di gestione ostacoli in stato non degradato.

In particolare, il chart è composto da 2 stati paralleli: *Sonars* e *B2Decisione*

1. ***B2Decisions***: Lo stato parallelo *B2Decisione* è dipendente dallo stato *Sonars* in quanto le sue transizioni vengono attivate da segnali provenienti da *Sonars*. In base ai segnali ricevuti, è capace di settare la



(a) Stato parallelo Sonars



(b) Stato parallelo B2Decisione

Figure 8: Stati paralleli del chart in stato non degradato.

variabile di output del chart, variabile che indica la decisione presa dal supervisore. Quindi, in questo stato si determina l'output del chart, che è un numero che varia da 0 a 4. Le azioni possibili includono l'arresto immediato del rover o la deviazione del percorso in base alla posizione dell'ostacolo. In quest'ultimo caso, la deviazione dura fintanto che il rover non ruota di 45° rispetto alla direzione iniziale, verso la direzione del sonar che per primo ha rilevato l'ostacolo.

2. ***Sonars***: All'interno di questo stato parallelo sono presenti altri *3* stati paralleli.

   (a) ***distance_70***: Rappresenta la condizione in cui uno dei sonar rileva un ostacolo a una distanza ≤ 70 cm.

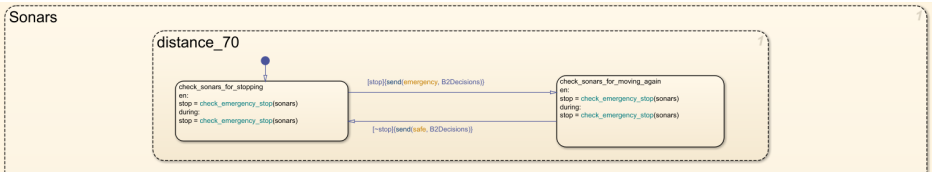


Figure 9: Stato parallelo per la gestione di un ostacolo a distanza ≤ 70 cm.

In questo stato quando uno dei sonar rileva un ostacolo a una distanza inferiore o uguale a 70 cm, viene attivata una transizione che porta allo stato di arresto immediato del rover. In particolare, quando un sonar rileva la presenza di un ostacolo a distanza ≤ 70 cm, viene inviato un segnale **Emergency** allo stato parallelo *B2Decisione* per fermare il rover. B2Decision utilizza questo

segnale per portarsi nello stato in cui l'output del chart prevede lo stop.

(b) ***distance_gt70 —- timers***: Questi due stati insieme permettono il rilevamento di un ostacolo in movimento tra le coppie di sonar

- *S1-S2* (tra sonar di sinistra e sonar centrale)
- *S2-S1* (tra sonar centrale e sonar di sinistra)
- *S2-S3* (tra sonar centrale e sonar di destra)
- *S3-S2* (tra sonar di destra e sonar centrale)

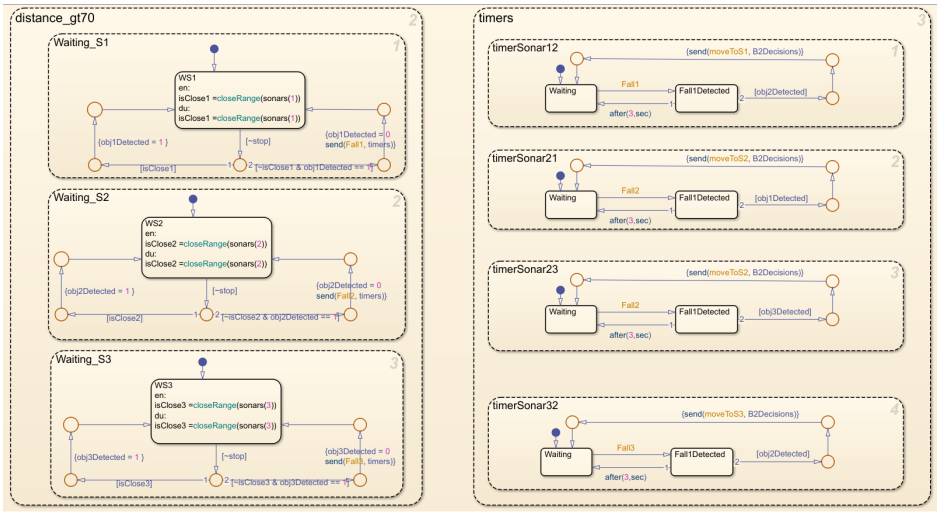


Figure 10: Stato parallelo per la gestione di un ostacolo in movimento a distanza > 70 cm.

Nello stato *distance_gt70* sono presenti 3 stati paralleli, *Waiting_S1*, *Waiting_S2*, *Waiting_S3*, uno per ogni sonar.

Di seguito si analizza la dinamica di rilevamento di un ostacolo che si sposta dal sonar $S1$ verso il sonar $S2$. Tale logica è da considerarsi valida per ogni coppia di sensori precedentemente elencata.

Si assume, come condizione necessaria, l'assenza di ostacoli a una distanza inferiore a 70 cm; in caso contrario, il sistema non procederebbe al rilevamento di oggetti in movimento.

i. **Attivazione** ($S1$): Quando il sonar $S1$ rileva un oggetto entro il range 100–300 cm, la variabile $obj1Detected$ viene impostata a 1 **(fronte di salita)**.

ii. **Transizione e Timing**: Nel momento in cui l'oggetto esce dal campo d'azione di $S1$, la variabile $obj1Detected$ torna a 0

(fronte di discesa). Contestualmente, lo stato timerSonar12 del modulo timers avvia un conteggio di 3 secondi.

iii. **Verifica ($S2$)**: Se il sonar $S2$ rileva l'ostacolo (sempre tra 100 e 300 cm) entro la finestra temporale dei 3 secondi, viene inviato il segnale $moveToS1$ allo stato parallelo $B2Decision$. Qualora il timer scada senza alcun rilevamento da parte di $S2$, non viene trasmesso alcun segnale.

2.4.2 Gestione ostacoli con sistema in stato *degradato*

Il chart per la gestione degli ostacoli in stato degradato è mostrato in Figura 11.

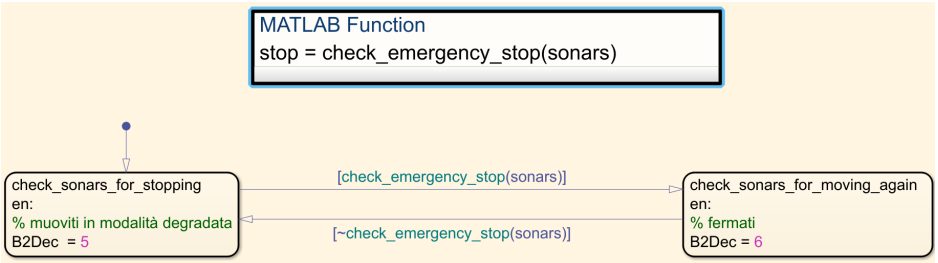


Figure 11: Chart di gestione ostacoli in stato degradato.

In questo caso, la logica di gestione degli ostacoli è semplificata rispetto allo stato non degradato. Infatti, l'unica condizione considerata è la presenza di un ostacolo a una distanza inferiore o uguale a 300 cm. Quando uno dei sonar rileva un ostacolo entro questo range, viene attivata una transizione che porta l'uscita del supervisore all'arresto immediato del rover.

3 Supervisore Board 1

3.1 Panoramica generale

Il supervisore della Board 1 è implementato come un modulo Simulink denominato **SupervisorB1**, il cui schema a blocchi è illustrato in Figura 12.

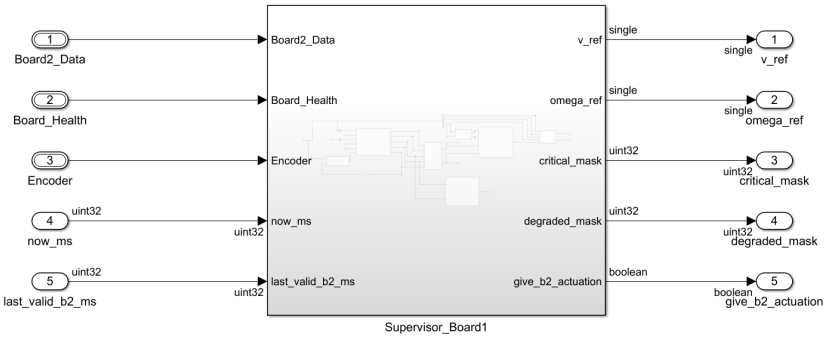


Figure 12: Schema a blocchi del modulo SupervisorB1.

Il suo compito è quello di decidere il riferimento di velocita (*v_ref*) e di direzione (*omega_ref*) del rover, in funzione dell'elaborazione dei dati di input. In particolare, esso è composto da 3 parti principali:

- **Gestione Faults**: si occupa di rilevare e gestire eventuali anomalie nei dati ricevuti dalla Board2, dagli encoder delle ruote, dai sensori di temperatura e batteria.
- **Decidere di far comandare la Board2**: decide se autorizzare o meno la Board2 a muovere il rover, in base alle condizioni di fault rilevate.
- **Aggregazione Fualt**: aggrega le anomalie rilevate nella parte di gestione faults e le codifica in due maschere di errore (critica e degradata).
- **Calcolo Riferimenti**: calcola i riferimenti di velocità lineare e angolare del rover in base ai comandi ricevuti dalla Board2 e alle condizioni di fault rilevate.

Nel seguito verranno descritti i segnali di input e output del supervisore, successivamente verranno descritte le tre parti principali del supervisore descritte sopra.

3.2 Input

I segnali in input che riceve sono:

- **Board2_Data**: rappresenta i dati provenienti dalla Board2. Quelli utilizzati dal supervisore sono:

– *command*: rappresenta il comando in uscita dal supervisore della Board2, che può assumere i seguenti valori:

```
typedef enum
{
  CMD_NORMAL = 0,
  CMD_ROTATE_180,
  CMD_GO_LEFT,
  CMD_GO_RIGHT,
  CMD_AVOID_RIGHT,
  CMD_AVOID_LEFT,
  CMD_STOP,
  CMD_ESTOP
} SupervisorCommand_t;
```

Figure 13: Comandi in uscita dal supervisore della Board2.

– *x_norm* & *y_norm*: rappresenta il comando utente proveniente dal joistick.

– *yaw*: rappresenta l'angolo di orientamento del rover, calcolato a partire dai dati provenienti dalla Board2.

– *imu_cohearence_status*: rappresenta la coerenza dei dati ricevuti dal sensore IMU, utile a rilevare motor fault.

– *critical_mask & degraded_mask*: sono due mashere di errore a 8 bit, in cui ogni bit indica la presenza di un'anomalia *critica* (da cui critical_mask) o *degradata* (da cui degraded_mask) specifica.

- **Board_Health**: rappresenta lo stato della Board1. La struttura dati è la seguente:

```
typedef struct
{
  float temperature_degC;
  float battery_pct;

  uint32_t task_last_run_ms;     /* ultima esecuzione del task */

  uint32_t temp_last_valid_ms;
  uint32_t batt_last_valid_ms;
} BoardHealthSnapshot_t;
```

Figure 14: Struttura dati Board_Health.

- **Encoder**: rappresenta i dati provenienti dagli encoder delle ruote del rover. La struttura dati è la seguente:

```
typedef struct
{
  float wheel_speed_rpm[4];
  bool hasNoFeedback[4];            // indica se c'è corrispondenza tra comando e lettura encoder.
                                    // (è Vero se la lettura encoder restituisce 0 rpm in presenza
                                    // di un comando di velocità valido)

  uint32_t task_last_run_ms;        /* ultima esecuzione del task */
  uint32_t data_last_valid_ms[4];   /* ultimo istante in cui i dati acquisiti sono validi
                                       (uno per ogni encoder) */
} EncoderSnapshot_t;
```

Figure 15: Struttura dati Encoder.

- **Now**: Rappresenta il tempo corrente in millisecondi.
- **Last_valid_b2_ms**: Rappresenta il tempo in millisecondi dell'ultimo dato valido ricevuto dalla Board2.

3.3 Output

I segnali in output che fornisce sono:

- **v_ref**: rappresenta il riferimento di velocità lineare del rover, in m/s.
- **omega_ref**: rappresenta il riferimento di velocità angolare del rover, in rad/s.
- **critical_mask & degraded_mask**: sono due mashere di errore a 8 bit, in cui ogni bit indica la presenza di un'anomalia *critica* (da cui critical_mask) o *degradata* (da cui degraded_mask) specifica, come descritto nella Tabella 1.

Bit	ID Segnale	Descrizione dell'Anomalia
0	TEMP_CRI/DEG	Errore termico (Logica predittiva)
1	BATT_CRI/DEG	Tensione batteria sotto soglia minima
2	COMM_CRI/DEG	Errore ricezione Board-to-Board
3-6	WHEEL_CRI/DEG	Guasti attuatori (FL, FR, RL, RR)
7	SUP_CRI/DEG	Timeout comunicazione Supervisore B2

Table 1: Mappatura della maschera di errore a 8 bit

- **give_b2_actuation**: rappresenta un segnale booleano che indica se la Board2 deve essere autorizzata a muovere il rover.

3.4 Gestione Faults

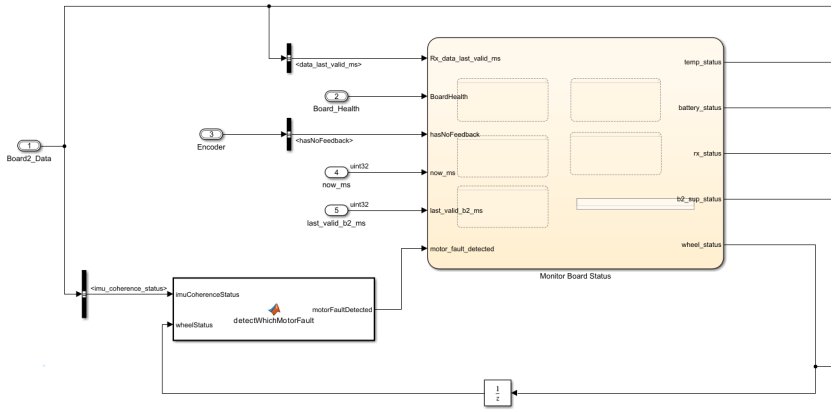


Figure 16: Chart di gestione dei faults.

L'output di questo chart è rappresentato da 5 segnali, ognuno dei quali indica la presenza o meno di un'anomalia *critica* o *degradata* specifica proveniente dalla lettura degli encoder, batteria, temperatura e comunicazione con la Board2. Di seguito si descrivono i valori che questi segnali possono assumere e le condizioni che portano a tali valori.

Table 2: Stati degradato e critico per Board 1

OUTPUT	**VALORE**	**SIGNIFICATO**
temp_status	TEMP_HEALTH_DEGRADED	Quando la temperatura è nel range $]-15;-5] \cup [55;+60[$
	TEMP_HEALTH_CRITICAL	• Quando la temperatura è per almeno 4s nel range $]-\infty;-15] \cup [60;+\infty[$ • Nel caso in cui l'intervallo di tempo dall'ultimo aggiornamento della temperatura è di 0.5s, si ipotizza che la temperatura aumenti di 1 °C/s. Se questa sale raggiungendo un valore superiore a 65 °C/s si ha questo valore.

OUTPUT	VALORE	SIGNIFICATO
	TEMP_HEALTH_OK	Quando la temperatura è nel range $]-5;55[$
batt_status	BATT_HEALTH_DEGRADED	Quando la percentuale di batteria è minore del 23%
	BATT_HEALTH_CRITICAL	• Quando la temperatura è per almeno 5 s nel range $[0\%;15\%]$ • Nel caso in cui l'intervallo di tempo dall'ultimo aggiornamento della percentuale batteria è di 0.5 s, si ipotizza che la percentuale diminuisca di 0.42%/s. Se questa diminuisce raggiungendo un valore minore a 15% si ha questo valore.
	BATT_HEALTH_OK	Quando la percentuale è $>$ 25%
wheel_status(i)	WHEEL_DEGRADED_ENCODER	Quando viene dato un riferimento di velocità ma gli rpm sono nulli per un certo periodo di tempo, infatti per impostare questo valore si considera anche l'inerzia delle ruote. Questa condizione è rilevata da $has\_no\_feedback(i)$.
	WHEEL_CRITICAL_MOTOR	Quando alla condizione degradata si aggiunge un'incoerenza del sensore IMU.

OUTPUT	VALORE	SIGNIFICATO
	WHEEL_OK	Quando non ci sono ne condizioni critiche ne degradate, imposta questo valore.
b2_sup_status	SUP_DEGRADED	Questo valore indica una discontinuità operativa della Board 2. Il supervisore della Board 1 monitora l'heartbeat del supervisore remoto, il quale incrementa il valore ogni volta che viene eseguito. Se la media degli ultimi 10 intervalli di aggiornamento superi i 40 ms, il sistema segnala uno stato di degrado del supervisore di Board 2.
	SUP_CRITICAL	Se l'intervallo di tempo dall'ultimo aggiornamento dell'heartbeat del supervisore di Board2 supera i 120 ms, imposta questo valore.
	SUP_OK	Quando non ci sono ne condizioni critiche ne degradate imposta questo valore.

OUTPUT	VALORE	SIGNIFICATO
rx_status	RX_DEGRADED	Questo valore identifica una comunicazione instabile. Sebbene il collegamento fisico sia attivo, fattori quali errori di checksum (CRC) o anomalie nella lunghezza dei pacchetti impediscono l'aggiornamento della variabile data_last_valid_ms. Il sistema monitora la qualità del link calcolando la media mobile degli ultimi 10 intervalli di ricezione valida; se tale media supera la soglia critica di 40 ms, viene segnalato il degrado della ricezione.
	RX_CRITICAL	Se l'intervallo di tempo dall'ultima ricezione corretta supera i 120 ms, imposta questo valore.
	RX_OK	Quando non ci sono ne condizioni critiche ne degradate imposta questo valore.

3.5 Decidere di far comandare la Board2

Il chart per decidere se autorizzare o meno la Board2 a muovere il rover è mostrato in Figura 17.

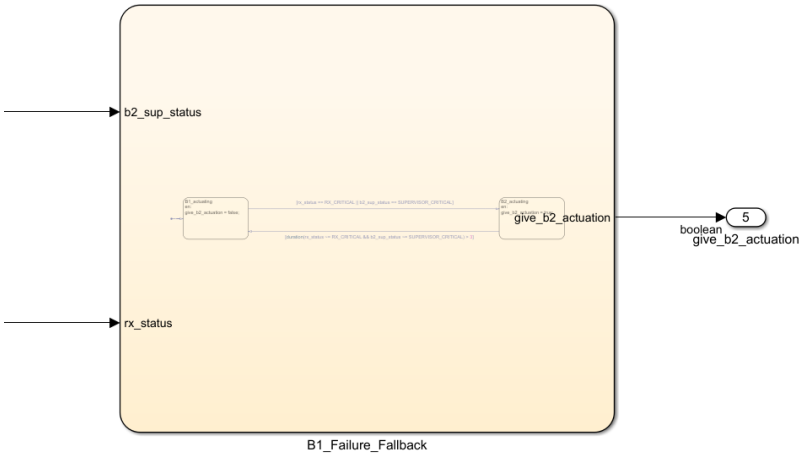
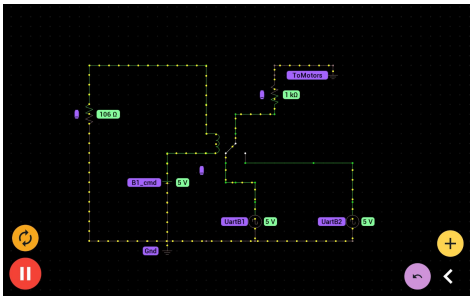


Figure 17: Chart per decidere se autorizzare o meno la Board2 a muovere il rover.

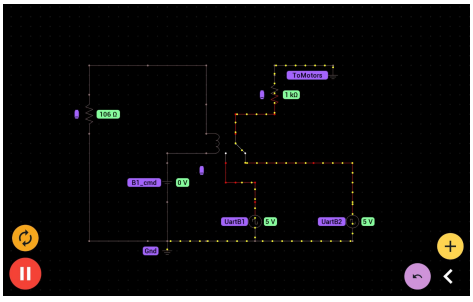
L'autorizzazione a Board 2 per il controllo del rover è regolata dalla variabile *give_b2_actuation* secondo la seguente logica:

- **give_b2_actuation = 1**: Il relè è aperto e il movimento viene gestito da Board 2. Questa condizione si ha quando sono riscontrate anomalie *critiche* in ricezione (texttt*rx_status = RX_CRITICAL), supervisore (textitb2_sup_status = SUP_CRITICAL).*
- ***give_b2_actuation = 0**: Il relè è chiuso e il movimento viene gestito da Board 1. Questa condizione si ha quando non sono riscontrate anomalie critiche in ricezione (textitrx_status = RX_OK), supervisore (textitb2_sup_status = SUP_OK).*

Tale meccanismo agisce come un dispositivo di sicurezza hardware basato sullo stato del supervisore, come mostrato nelle Figure 18b e 18a.



(a) Il comando di Board1 impedisce a Board2 di muovere il rover.



(b) Board1 permette alla Board2 di muovere il rover.

3.6 Costruzione maschere degradate e critiche

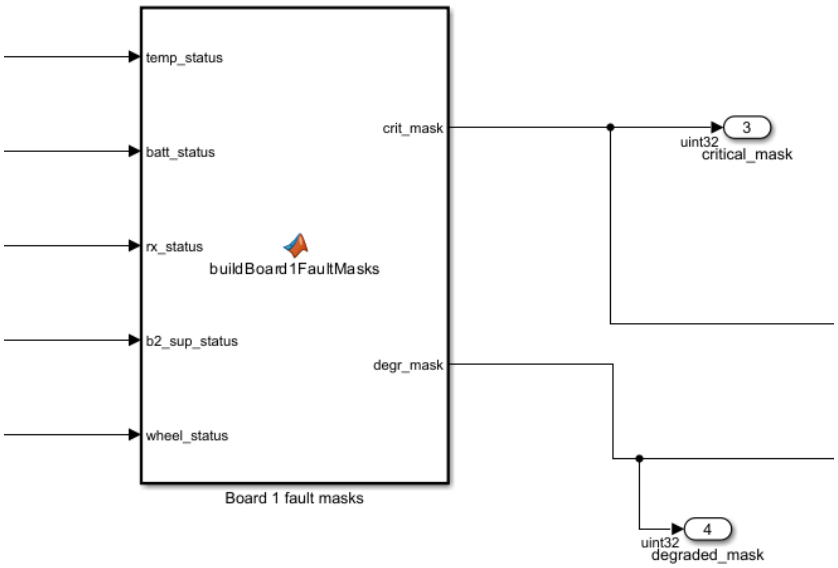


Figure 19: Chart per la costruzione delle maschere di errore critiche e degradate.

Dagli stati critici e degradati rilevati nella parte di gestione faults, si costruiscono due maschere di errore a 8 bit, una per le anomalie critiche e una per quelle degradate.

Il bit di ogni maschera rappresenta un'anomalia specifica, come descritto nella Tabella 1.

Bit	**Componente**	**critical_mask**	**degraded_mask**
0	Temperatura	1 se **TEMP_HEALT_CRITIC**	1 se **TEMP_HEALT_DEG**
1	Batteria	1 se **TEMP_HEALT_CRITIC**	1 se **TEMP_HEALT_DEG**
2	Ricevitore (RX)	1 se **RX_CRITICAL**	1 se **RX_DEGRADED**
3	Ruota FL	1 se **TEMP_HEALT_CRITIC**	1 se **TEMP_HEALT_DEG**
4	Ruota FR	1 se **TEMP_HEALT_CRITIC**	1 se **TEMP_HEALT_DEG**
5	Ruota RL	1 se **TEMP_HEALT_CRITIC**	1 se **TEMP_HEALT_DEG**
6	Ruota RR	1 se **TEMP_HEALT_CRITIC**	1 se **TEMP_HEALT_DEG**
7	B2 Supervisor	1 se **TEMP_HEALT_CRITIC**	1 se **TEMP_HEALT_DEG**

Table 3: Mappatura dei Bit nelle Fault Masks

3.7 Calcolo riferimenti

Il chart per il calcolo dei riferimenti di velocità lineare e angolare del rover è mostrato in Figura 20.

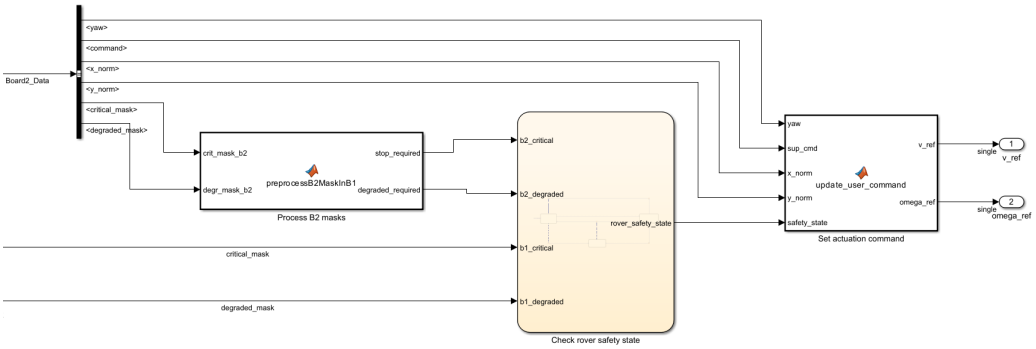


Figure 20: Chart per il calcolo dei riferimenti di velocità lineare e angolare del rover.

In questo chart, i riferimenti di velocità lineare (v_ref) e angolare ($omega\_ref$) del rover vengono calcolati in base ai comandi ricevuti dalla Board2 e alle condizioni di fault rilevate. In particolare, se non sono presenti anomalie critiche, incluse quelle della Board 2, i riferimenti vengono calcolati in base ai comandi ricevuti dalla Board2, con una logica di scaling che tiene conto della posizione del joystick e dell'angolo di orientamento del rover. Se invece sono presenti anomalie critiche, i riferimenti vengono impostati a 0, fermando il rover, indipendentemente dai comandi ricevuti dalla Board2.

La funzione `update_user_command` implementa un'arbitrazione dei comandi basata su livelli di priorità. Il modello cinematico segue una logica differenziale dove i riferimenti $(\vec{v}, \vec{\omega})$ sono definiti come:

$$\begin{cases} v_{ref} = y_{norm} \cdot V_{max} \cdot K_{safety} \\ \omega_{ref} = x_{norm} \cdot \Omega_{max} \cdot K_{safety} \end{cases} \tag{1}$$

dove $K_{safety} \in \{0, 0.5, 1\}$ è il coefficiente di riduzione derivante dallo stato di sicurezza.

3.7.1 Manovra di Inversione Automatica

Il sistema rileva un'intenzione di inversione quando $y_{norm} < -0.6$. In questa fase, il controllo passa da *User-in-the-loop* a *Automatic Heading Control*,

dove la velocità angolare è regolata dall'errore di puntamento θ_ϵ:

$$\omega_{ref} = f(\theta_{yaw} - \theta_{target}) \tag{2}$$

Il controllo termina quando $|\theta_\epsilon| < 12°$.