UNIVERSITÀ DI BRESCIA
FACOLTÀ DI INGEGNERIA
Dipartimento di Elettronica per l'Automazione

Laboratorio di Robotica Avanzata
Advanced Robotics Laboratory

Corso di Robotica
(Prof. Riccardo Cassinis)

Sistemazione software per Survivor 3

Elaborato di esame di: **Davide Belleri, Federico Lucchini**

Consegnato il: **22 settembre 2003**

Sommario

Lo scopo del nostro elaborato è di correggere e rendere funzionante il software realizzato da Alessandro Zanacchi per la movimentazione di Survivor 3. Abbiamo realizzato una nuova versione sempre in linguaggio assembler per il PIC 16F876 senza modificare la parte hardware della scheda elettronica. Le modifiche al programma originale sono state apportate seguendo le specifiche del Prof. Riccardo Cassinis e i suggerimenti dell'artista Laura Morelli.

1. Introduzione

Survivor 3 è la terza versione di una creazione artistica il cui scopo è di sensibilizzare le persone sul dramma delle mine antiuomo. In particolare sono messi in risalto le difficoltà, sia a livello motorio sia psicologico, di una persona vittima di tali ordigni. Tutto ciò attraverso una sedia che 'vede' gli oggetti e le persone che la circondano tramite sensori e 'cammina' in mezzo a loro grazie a due gambe anteriori mobili azionate da motori.

La versione del programma da cui siamo partiti gestisce tramite una 'subsumption architecture' tutti i possibili movimenti della sedia. Per emulare il comportamento di un essere vivente il sistema di controllo riceve in ingresso anche 'stimoli interni' che lo informano sullo stato in cui si trova. Survivor grazie agli 'stimoli esterni' riesce ad esplorare l'ambiente che la circonda evitando gli ostacoli e scappando da eventuali pericoli, grazie agli 'stimoli interni' modula la velocità con cui muoversi.

2. I problemi affrontati

2.1. Panico iniziale

Il primo problema che si nota nel provare Survivor con il software di Zanacchi è che all' accensione la sedia comincia a muoversi con velocità di panico cioè la velocità massima consentita. Questo agli occhi dell'artista Laura non appare corretto poiché la situazione di panico deve entrare in gioco solo quando un ostacolo si avvicina ad alta velocità e la sedia rileva un potenziale pericolo.

Il comportamento a prima vista bizzarro della sedia è dovuto al fatto che all' accensione tutte le distanze degli ostacoli vengono poste via software a 0 prima di leggere ciò che realmente rilevano i sensori. Così facendo la persona che accende la sedia, trovandosi in prossimità di essa, è sentita dai sensori. Per come è effettuato il calcolo della velocità degli oggetti, valore nuovo del sensore meno valore vecchio (0), risulta che la persona si è avvicinata molto velocemente.

Per ovviare a questo problema abbiamo introdotto un ritardo di circa 3 secondi tra l'attivazione dell'interruttore e l'istante in cui iniziano le conversioni dei sensori in modo che l'operatore abbia tempo per allontanarsi da Survivor. Quello che ci ha fatto intraprendere questa via di soluzione è stato il fatto che l'introduzione di un ritardo iniziale non porta ad una modifica della percezione del funzionamento di Survivor.

2.2. Posizionamento iniziale

Quando Survivor viene spento mentre sta facendo un passo i motori si fermano nella posizione corrente che spesso non corrisponde con la loro posizione di riposo. La conseguenza è che appena verrà riaccesa, la sedia si troverà piegata su un lato. Se siamo in una fase di esplorazione in cui la sedia cammina evitando gli ostacoli uno spettatore esterno non nota il piegamento iniziale, mentre se all'accensione siamo in fase di standby la sedia rimarrà in questa posizione finchè non vede un oggetto o scade il tempo di standby. Questo stato iniziale se prolungato nel tempo può indurre l'osservatore a pensare ad un cattivo funzionamento di Survivor.

2.3. Completamento passi a bassa velocità

A basse velocità di funzionamento la coppia fornita dai motori può non risultare sufficiente per completare un passo. La sedia inizia in modo corretto il movimento della gamba ma quando arriva a contatto con il terreno non riesce a riportarsi nella posizione di riposo ma resta bloccata. Questa situazione permane finché la sedia non viene fisicamente aiutata a rialzarsi. Se questa ultima operazione avviene tardivamente può provocare il surriscaldamento del motore bloccato che continua ad essere alimentato.

2.4. Funzionamento su diversi terreni

Provando Survivor su due terreni diversi, entrambi possibili luoghi di esposizione (cemento e pavimento in piastrelle), abbiamo subito notato come il set di velocità di funzionamento adatto in un ambiente risulti inutilizzabile nell'altro. In particolare a basse velocità dei motori mentre sulle piastrelle la sedia riesce a muoversi sul cemento la coppia fornita dai motori non è sufficiente a completare i passi.

2.5. Gestione situazioni critiche

La sedia è fornita di 8 sensori come illustrato nella figura 1

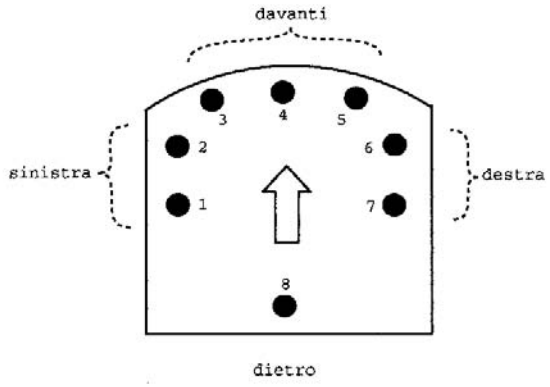

Figura 1 Disposizione dei sensori sul piano della sedia

La precedente versione del programma si basa sulla seguente logica:

1. un ostacolo laterale per influenzare il movimento di Survivor deve essere rilevato come fonte di pericolo cioè in veloce avvicinamento o in concomitanza con un ostacolo frontale. Se la seggiola non 'vede' niente davanti procede in direzione rettilinea ignorando ciò che gli sta sui lati.

2. Se viene rilevato un ostacolo vicinissimo davanti la sedia si blocca e suona il buzzer fino a quando l'ostacolo sparisce e ricomincia quindi il normale funzionamento. Lateralmente e dietro è possibile avvicinarsi lentamente, per non provocare una situazione di panico, fino a toccare i sensori e la sedia, senza influire sul suo funzionamento.

Entrambi questi comportamenti sono letti da uno spettatore come mancanza di sensibilità della sedia verso il mondo esterno e rendono possibile il contatto con parti delicate di Survivor senza darne segnalazione di pericolo con il suono del buzzer.

2.6. La gestione dei motori

La gestione dei motori è funzionale ma poco elegante.

I motori utilizzati sono dotati di un interruttore atto a commutare lo stato logico di un ingresso del PIC ogni volta che viene completato un giro. Quando il motore è in moto sul piedino del PIC vengono rilevati circa 5V, mentre se viene concluso un giro passa a circa 0V. Il problema è che quando il motore riparte il livello del piedino passa nuovamente a 5V. Senza una gestione adatta del PIC, viene generata un'interruzione ogni volta che vi è un cambiamento di stato sulla linea e quindi risulta necessario trovare un modo per scartare la prima interruzione generata alla partenza di ogni giro. Inoltre l'accensione e lo spegnimento del motore sono controllati invertendo la direzione (input o output) di PORTC,1 e PORTC,2, operazione secondo noi non necessaria.

2.7. Lo sbilanciamento

Se la sedia si muove ad alte velocità succede che, a causa della posizione dei vari elementi che la costituiscono, si sbilancia dalla parte sinistra fino quasi a cadere. Il problema è che lo sbilanciamento avviene in modo sistematico rendendolo prevedibile e conferendo a Survivor un atteggiamento indesiderato.

2.8. Impostazione e assegnazione delle velocità ai motori

Analizzando il funzionamento di Survivor abbiamo visto come le velocità impostate dal programma precedente siano inadeguate poiché variabili tra il 30% e il 90% della velocità massima del motore. Dopo alcune prove abbiamo osservato come per essere sicuri che il motore compia un giro completo dobbiamo assegnargli una velocità non inferiore al 70%. Il modo in cui calcolare le velocità da assegnare ai motori è quindi stato completamente rivisto.

2.9. Collegamento seriale (RS232)

La parte di programma riguardante la comunicazione via seriale utilizza una gestione ad interrupt permettendo al programma di continuare l'esecuzione nei tempi in cui si attende l'invio di dati da terminale verso la scheda.

Tuttavia essa risulta non funzionante e, dal punto di vista della programmazione e della documentazione, alquanto incomprensibile. Inoltre procede alla scrittura in eeprom ogni volta che si accende la sedia (vengono memorizzate le costanti utilizzate nel programma) o su apposito comando andando a scrivere i valori letti dai sensori (operazione alquanto inutile). Tutto ciò è da evitare poiché la memoria eeprom si degrada se scritta in modo ripetitivo. È anche necessario implementare controlli diagnostici per verificare il corretto funzionamento del programma.

3. La soluzione adottata

3.1. Posizionamento iniziale

Per far sì che all'avvio della sedia le gambe si portino nella posizione di riposo abbiamo introdotto una parte di programma all'interno della procedura 'configPWM' che viene eseguita ad ogni avvio. Prima di abilitare gli interrupt provenienti dagli interruttori montati sui motori viene controllata con una gestione da programma la posizione attuale dei motori: in particolare se il motore sinistro si trova in posizione di riposo si passa ad analizzare l'altro motore. Se anche questo risulta correttamente posizionato il programma continua il suo flusso normalmente. Nel caso in cui uno dei due motori o entrambi siano fuori dalla posizione di riposo (rilevabile testando i livelli dei segnali sui piedini PORTB,6 e PORTB,7) si procede prima per il motore sinistro poi per il destro a fornire un'elevata velocità di rotazione (circa 90% della velocità massima) settando i registri CCPR1L e CCPR2L e poi continuando a controllare ciclicamente se scatta l'interruttore di fine giro.

La scelta di una velocità così elevata per effettuare questo movimento è dettata dal fatto che un eventuale bloccaggio del motore dovuto al tipo di terreno non è gestito come invece accade durante il funzionamento post-configurazione.

3.2. La gestione dei motori

Per la gestione degli interrupt provenienti dai motori abbiamo provato a procedere come indicato dall'application note AN566 cioè considerando solo le variazioni da

alto a basso (fine giro) degli interruttori collegati ai motori ed ignorando quelle da basso ad alto (inizio giro).

A fronte di una semplificazione delle modalità di controllo (eliminazione della variabile 'contamotore') si è presentato un problema inaccettabile per il corretto funzionamento della sedia. Infatti facendo girare i motori abbiamo visto che in modo casuale delle volte viene gestita solo l'interruzione desiderata mentre altre volte vengono gestite ambedue le interruzioni come eventi di fine giro. Il fatto che questo comportamento vari in modo casuale ci ha indotto a pensare che non ci sia un errore di programmazione, ma che il tutto sia causato da spikes indesiderati presenti sulle linee degli interruttori. Questa idea è nata vedendo che sulla stessa scheda elettronica sono presenti sia i circuiti di segnale sia i MOS di potenza per i motori. In conclusione abbiamo pensato di mantenere la struttura preesistente per la gestione degli interrupt dei motori in quanto funzionante e partire da questa come base per risolvere i problemi esposti nel capitolo precedente.

Innanzi tutto abbiamo cambiato le modalità di accensione/spegnimento dei motori; queste sono ora possibili scrivendo direttamente nei registri CCPR1L e CCPR2L il valore del duty-cycle del PWM. Quindi se è necessario fermare un motore in modo istantaneo è sufficiente settare a 0 il registro corrispondente senza dover invertire il verso di funzionamento di PORTC.

3.3. Funzionamento su diversi terreni

Per fornire alla sedia una sorta di intelligenza e renderla capace di muoversi senza difficoltà su diversi terreni abbiamo realizzato la procedura 'tasta_terreno'. Questa viene eseguita ogni volta che viene avviato il programma e attraverso una sequenza di passi a velocità crescenti riesce a individuare la velocità minima di rotazione dei motori per completare un giro.

In particolare essa viene chiamata dopo la sezione di configurazione iniziale del PIC e come prima operazione disabilita temporaneamente gli interrupt provenienti dal convertitore A/D e dal Timer1. Attraverso la variabile di supporto 'tasto', il programma può esclusivamente gestire interruzioni provenienti dai motori (PORTB) e ignorare il flag del Timer1 che comunque continua a funzionare senza generare interruzioni. Il motore su cui vengono effettuate le varie prove è il sinistro poiché per come è montato compie uno sforzo maggiore per eseguire un passo. La prima velocità assegnata al motore è 2FH (75% della velocità massima). Se la procedura 'intportb' rileva il completamento del giro prima che sia passato un tempo pari a TEMPO_MAX (circa 5 secondi) allora, sempre in quest'ultima procedura, la velocità utilizzata viene incrementata di 2 e poi 4 e salvata in eeprom rispettivamente come 'velfermi' e 'velmoto'. Sommare i valori 2 e 4 fornisce un margine di sicurezza garantendo il completamento del giro anche in condizioni un po' peggiori delle attuali (il pavimento infatti può presentare delle disomogeneità). Il valore in 'velfermi' è la velocità minima di funzionamento della sedia, mentre 'velmoto' contiene la velocità a cui la sedia ricomincia a camminare dopo essersi prima fermata (viene creato un ciclo di isteresi). Trovata la velocità adatta il programma resetta 'tasto', riabilita gli interrupt e continua il regolare flusso. Se invece la gamba sinistra non riesce a fare un giro in TEMPO_MAX essa viene riportata a velocità massima nella posizione di riposo e si ritenta di completare il giro con una velocità superiore del 5% rispetto alla precedente. Il valore di TEMPO_MAX viene incrementato attraverso un polling sul

flag di Timer1. La variabile 'limite' svolge in questa procedura lo stesso ruolo di 'contamotore'.

Procedure coinvolte: 'intportb', 'tasta_terreno'

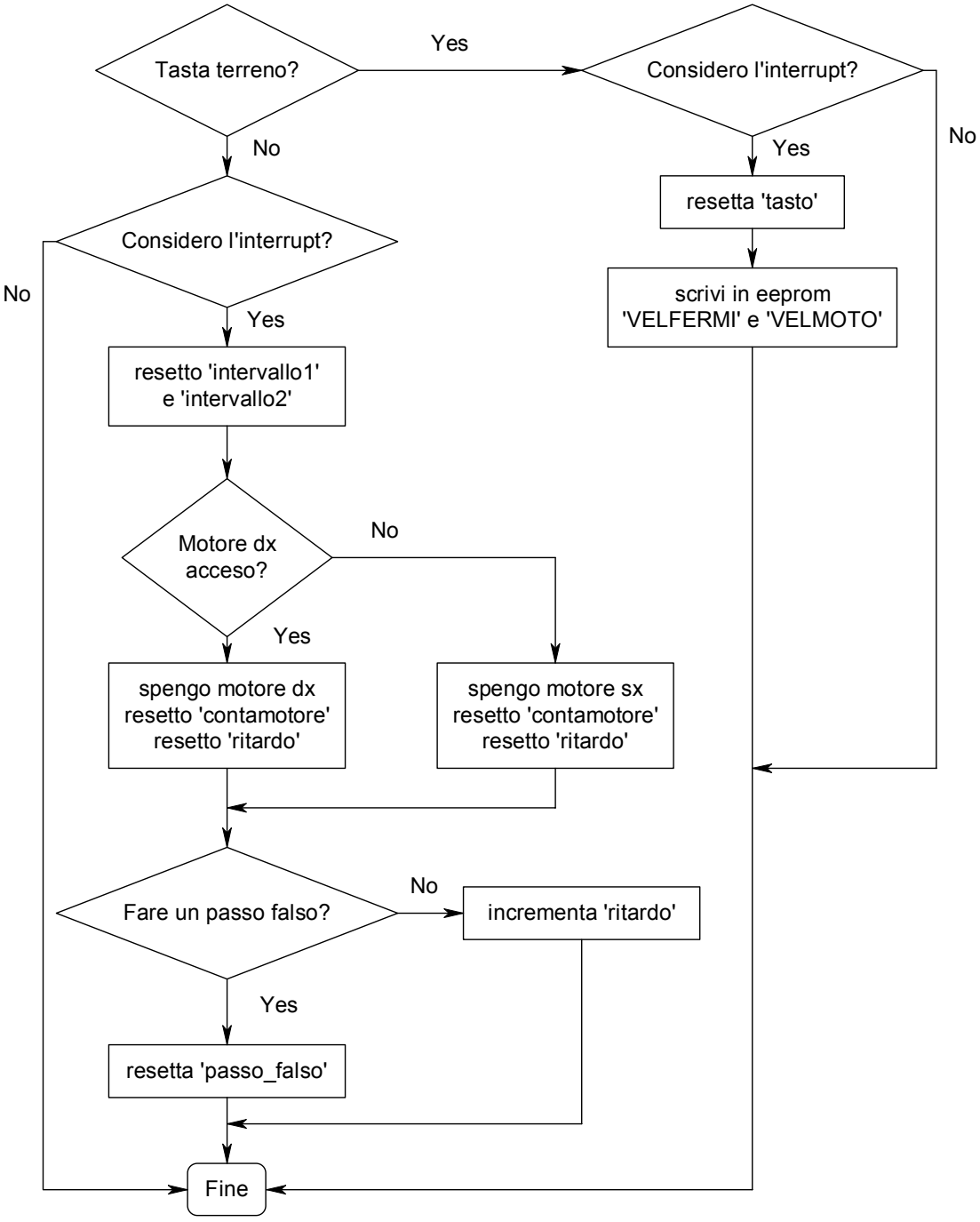

Figura 2 Diagramma di flusso della procedura 'intportb'

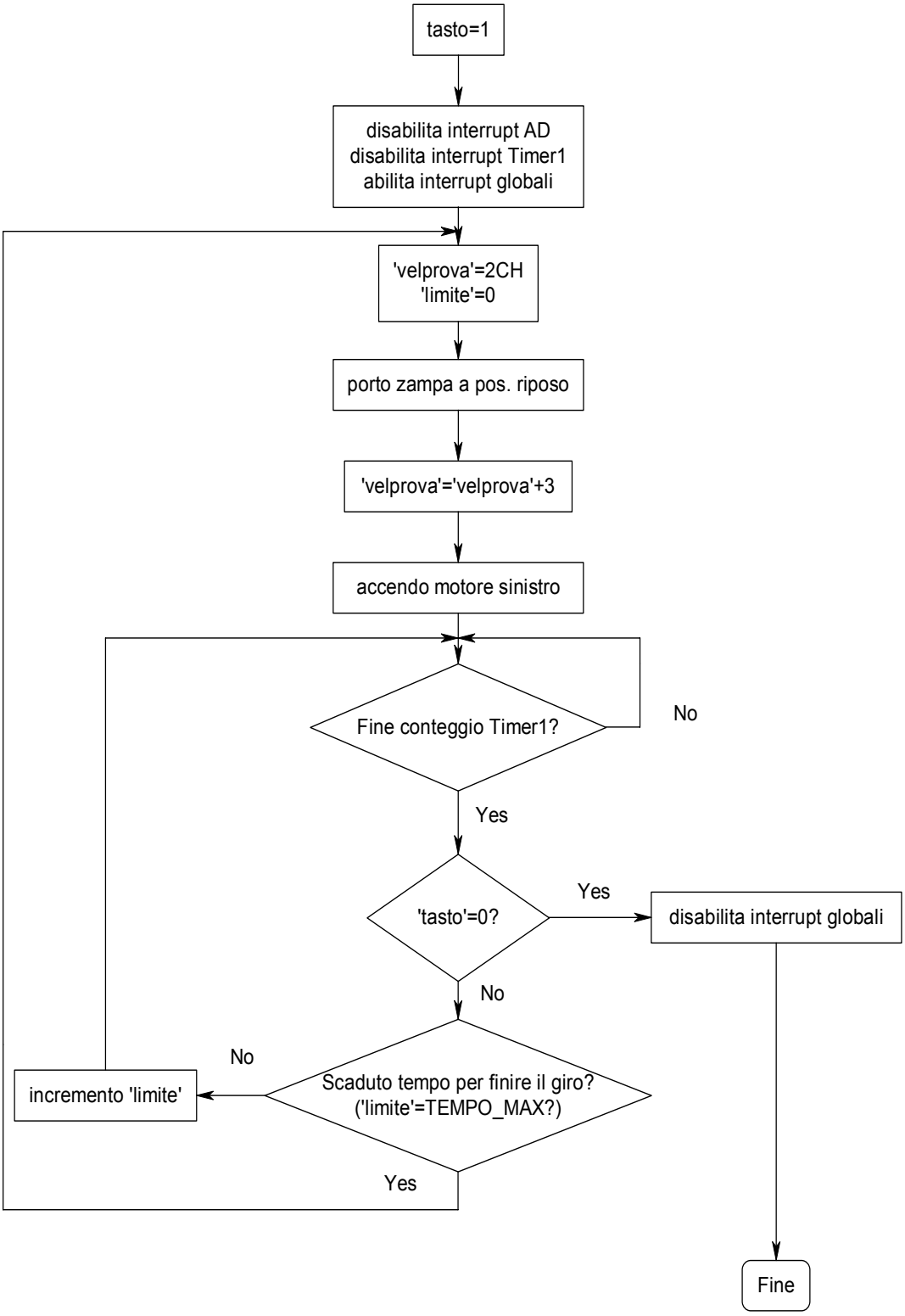

Figura 3 Diagramma di flusso della procedura 'tasta_terreno'

3.4. Completamento passi a bassa velocità

L'unico modo a disposizione del PIC per verificare il bloccaggio di un motore (dovuto o a coppia insufficiente o a fattori esterni) è controllare da quanto tempo sta comandando la sua alimentazione. Per identificare queste situazioni e poterle discriminare tra loro è stato implementato un sistema basato su due variabili che fungono da contatori ('intervallo1' e 'intervallo2').

Quando un motore inizia un giro ad ogni interruzione del 'Timer1' (65 ms) viene incrementata 'intervallo1'. Se arriva un'interruzione dal PORTB interpretata come fine giro 'intervallo1' viene resettata e viene fermato il motore corrispondente. Nel caso in cui 'intervallo1' raggiunge il valore 64H (pari a circa 6,5 secondi) significa che il motore non è riuscito a completare il giro con la velocità assegnata e quindi nella procedura 'esegui' viene data al motore la nuova velocità 39H (90% della velocità massima). Il programma tramite il contatore 'intervallo2' (anch' esso incrementato dal 'Timer1' ogni 65 ms) controlla se viene terminato il giro entro 2 secondi dall'assegnazione della nuova velocità (corrispondenti al valore 20H di 'intervallo2'). In tal caso vengono azzerati i due contatori e il motore fermato nella posizione di riposo. Se nonostante la velocità superiore imposta il motore non segnala che ha finito il giro, molto probabilmente un fattore esterno o il malfunzionamento del motore non consente alla gamba di muoversi e si procede quindi in modo precauzionale ad interrompere il movimento del motore togliendone l'alimentazione e a segnalarne il possibile guasto facendo suonare il buzzer in modo continuo. Questo comportamento è ottenuto mediante un ciclo ripetuto infinite volte con l'attenzione di avere disabilitato gli interrupt generali e isolando così la sedia dagli stimoli esterni. Da questa situazione di blocco è possibile uscire solo attraverso lo spegnimento e la riaccensione della sedia.

Procedure coinvolte: 'intportb', 'intTimer1', 'esegui'.

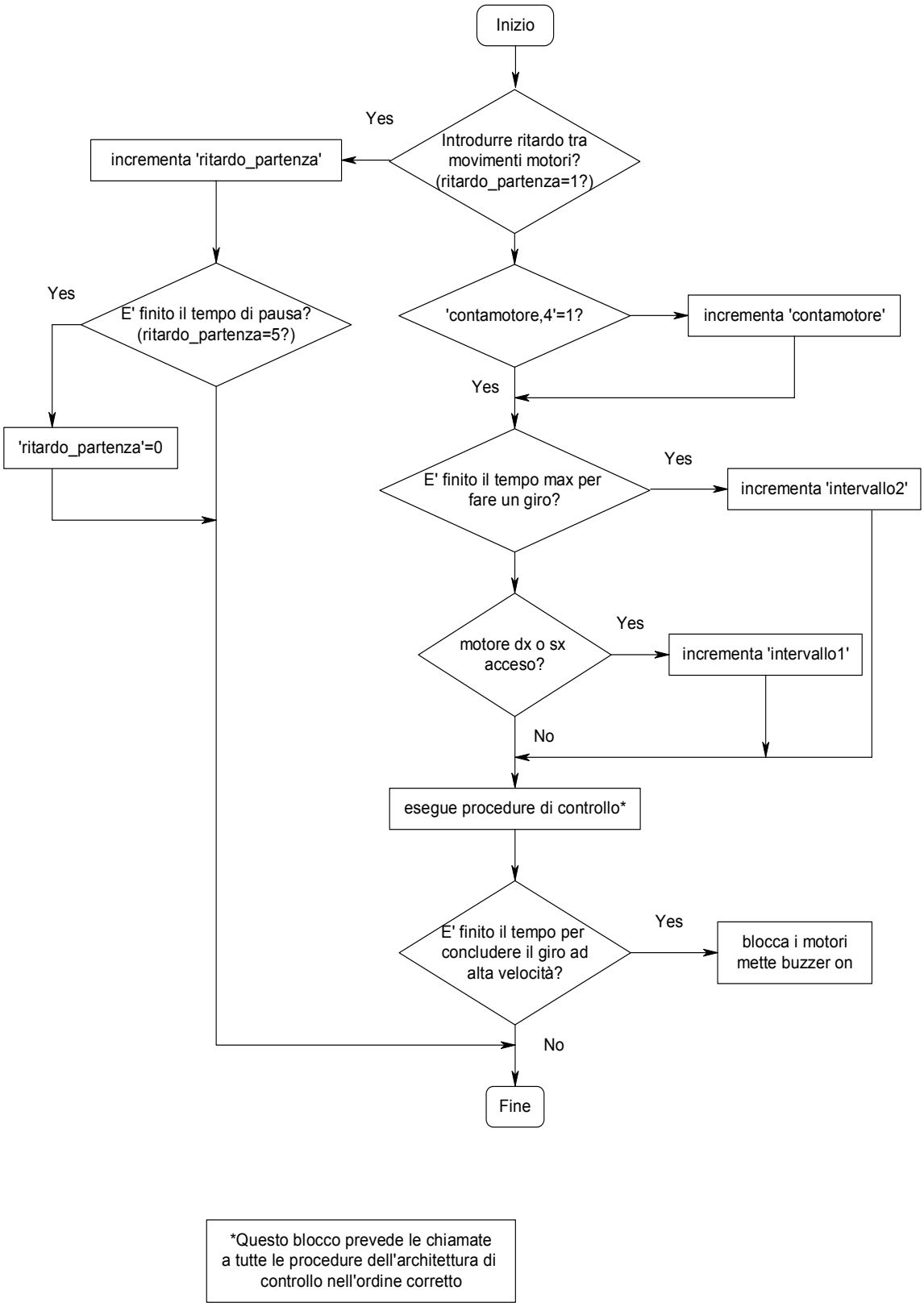

Figura 4 Diagramma di flusso della procedura 'intTimer1'

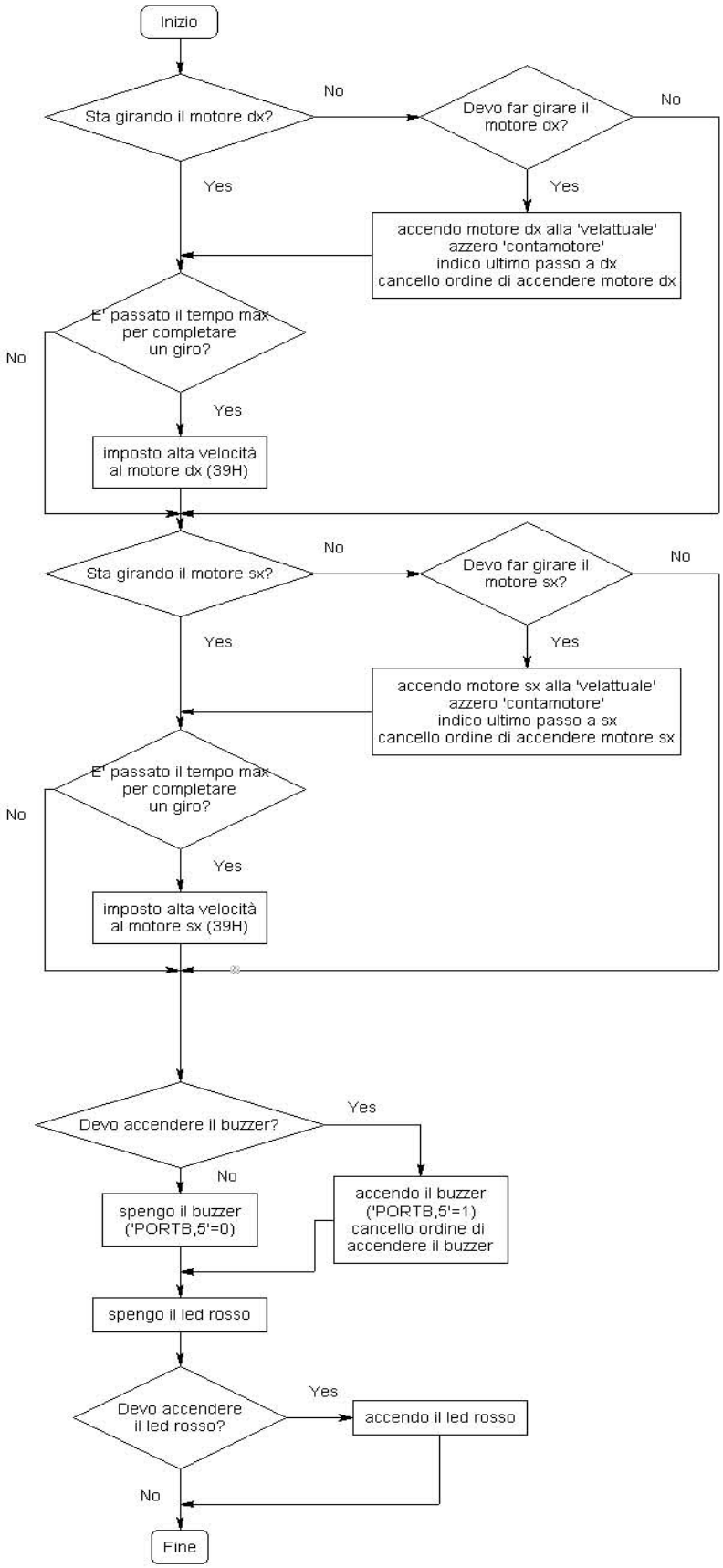

Figura 5 Diagramma di flusso della procedura 'esegui'

3.5. Lo sbilanciamento

Facendo passi consecutivi senza pause tra la fine dell'uno e l'inizio del successivo entra prepotentemente in gioco la dinamica associata al sistema sedia più pesi. A fronte della complessità di questo problema meccanico (presente anche a velocità inferiori rispetto alla velocità massima di funzionamento dove però ha frequenza minore ed è meno accentuato) abbiamo pensato di introdurre tra la fine di un passo e l'inizio dell'altro un ritardo temporale pari a 0,3 secondi, che viene calcolato tramite l'ausilio del contatore 'ritardo_partenza', al fine di ricondursi ad una più semplice analisi di tipo cinetostatico.

Come già detto precedentemente il problema di questo sbilanciamento è la prevedibilità e l'eccessiva frequenza con cui accade. Infatti se avviene con frequenza simile alle cadute risulta molto apprezzato dall'artista Laura. La casualità di questo comportamento è il risultato della combinazione tra l'eliminazione del ritardo tra un passo e l'altro ad una frequenza costante, l'insieme di parametri relativi allo stato di funzionamento e l'intervento di elementi esterni (ad esempio: inclinazione del terreno, passi alternati, passi sempre con la stessa gamba, posizione piombi, ...). La frequenza con cui viene tolto il ritardo tra passi successivi è pari alla frequenza con cui la variabile 'casuale' assume un valore multiplo di 3. Ricordiamo che 'casuale' dopo essere generata nella procedura 'tempocasuale' viene decrementata fino al valore zero ogni 8 secondi e poi viene ricreata.

Durante le pause tra i movimenti delle gambe non si bloccano le conversioni dei sensori. Le cadute sono comandate quando la variabile 'casuale' assume il valore 06H e quindi si verifica sempre una caduta associata a sbilanciamento, anch'essa valutata in modo positivo dall'artista.

Il meccanismo che regola la caduta è il seguente: un motore viene messo in moto mentre l'altro è già in movimento e la mancanza di sincronia tra i momenti di avvio dei motori permette di avere diverse tipologie di caduta. Inoltre possono accadere cadute consecutive a causa della gestione delle interruzioni provenienti dai motori attraverso la variabile 'contamotore'.

Procedure coinvolte: 'tempocasuale', 'ognisedici', 'intTimer1', 'intportb'

3.6. Impostazione e assegnazione delle velocità ai motori

Attraverso 'tasta_terreno' vengono memorizzati in eeprom i valori 'velfermi' e 'velmoto'. In base a questi il nostro programma genera due range di velocità, uno per quanto riguarda la 'fame' cioè il livello di carica della batteria e l'altro per quel che riguarda la 'fatica' cioè il tempo da cui la sedia è in continuo movimento.

Il primo range varia tra 'velfermi'+1 e 'velfermi'+7. Il primo valore corrisponde allo stato di batteria quasi scarica mentre il secondo a quello di carica completa. È da sottolineare come alla sedia venga assegnata velocità nulla quando il livello della batteria risulta minore di 'troppafame'. Raggiunto questo livello si osserva l'inizio del lampeggio del led rosso che segnala batteria scarica. Tuttavia, prima di fermarsi in modo definitivo, i motori potrebbero continuare a girare per un po' di tempo (una ventina di secondi) per oscillazioni nel livello della batteria causate dall'accensione/spegnimento dei motori stessi. Se la batteria risulta scarica e viene generata una situazione di panico la sedia comunque corre con il led lampeggiante.

Potrebbe succedere che su un terreno molto accidentato 'velfermi' venga impostata ad un valore molto elevato da 'tasta_terreno'. In tal caso il range viene limitato superiormente dal valore 3FH che corrisponde alla massima velocità (100%) di rotazione dei motori.

Il range di velocità riguardante la 'fatica' varia da 'velfermi'-1 a 'velfermi'+6. Il livello inferiore così stabilito permette alla sedia di fermarsi a causa della fatica. Discorso analogo a quello fatto per la 'fame' vale quando 'velfermi' viene impostata ad un valore molto elevato.

Calcolati i valori di 'velfame' e 'velfatica' rispettivamente nelle procedure 'fame' e 'fatica' in 'velocitam' si controlla quale delle due è da imporre ai motori. Il controllo del limite 'velfermi' è effettuato solo sulla variabile 'velfatica'.

Procedure coinvolte: 'fame', 'fatica', 'velocitam'.

velprova	VELFERMI	Minimo velfatica	Massimo velfatica
2FH	31H	30H	37H
32H	34H	33H	3AH
35H	37H	36H	3DH
38H	3AH	39H	**3FH**

Tabella 1 Limiti del range di variazione di 'velfatica' in funzione di VELFERMI

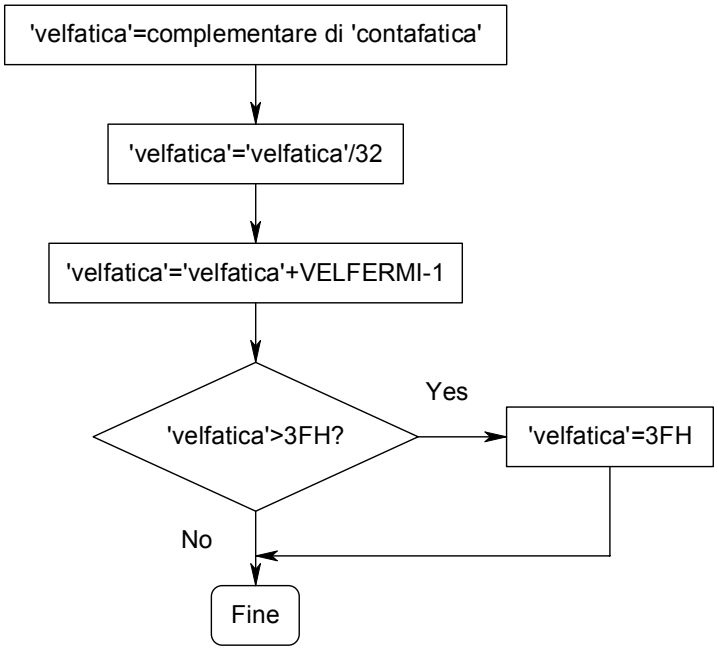

Figura 6 Diagramma di flusso della procedura 'velfatica'

velprova	VELFERMI	Minimo velfame	Massimo velfame
2FH	31H	32H	38H
32H	34H	35H	3BH
35H	37H	38H	3EH
38H	3AH	3BH	**3FH**

Tabella 2 Limiti del range di variazione di 'velfame' in funzione di VELFERMI

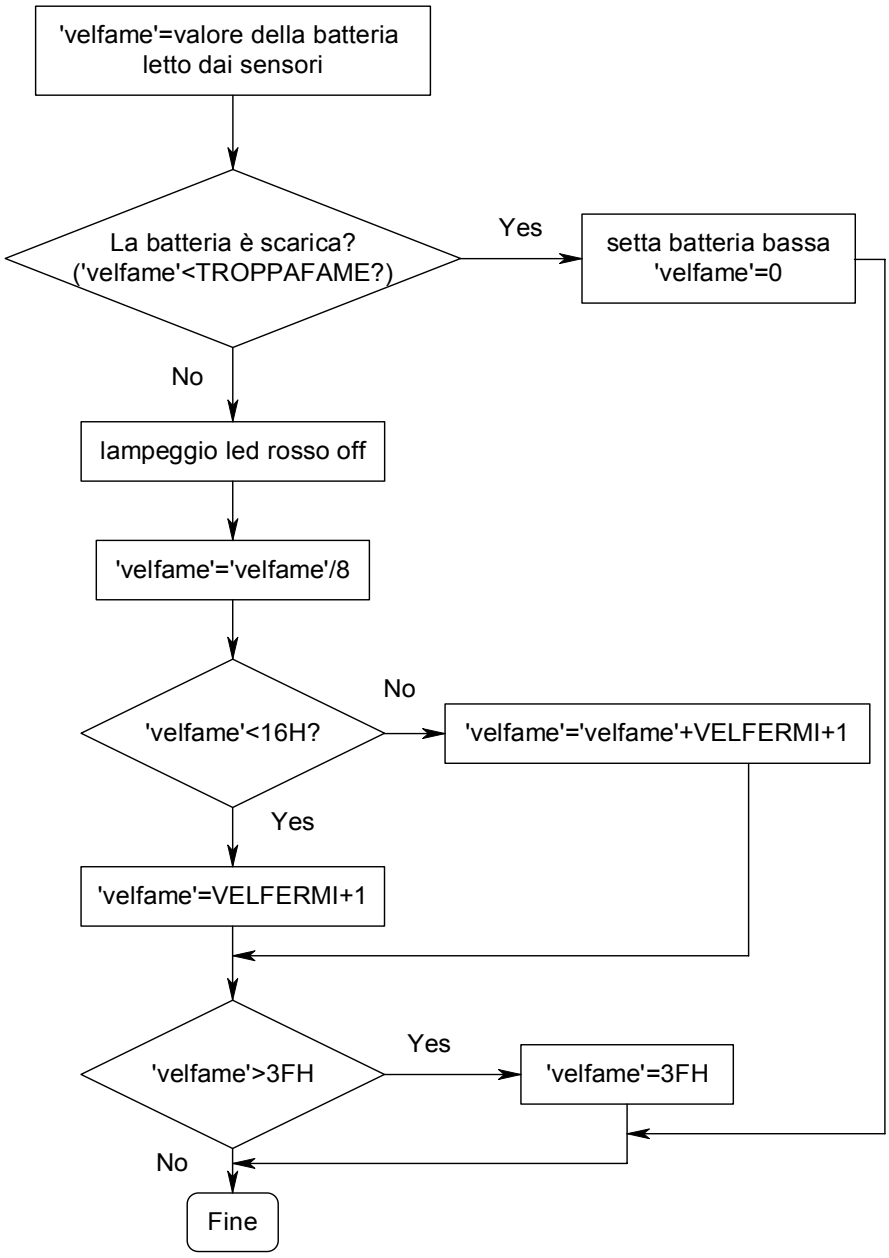

Figura 7 Diagramma di flusso della procedura 'velfame'

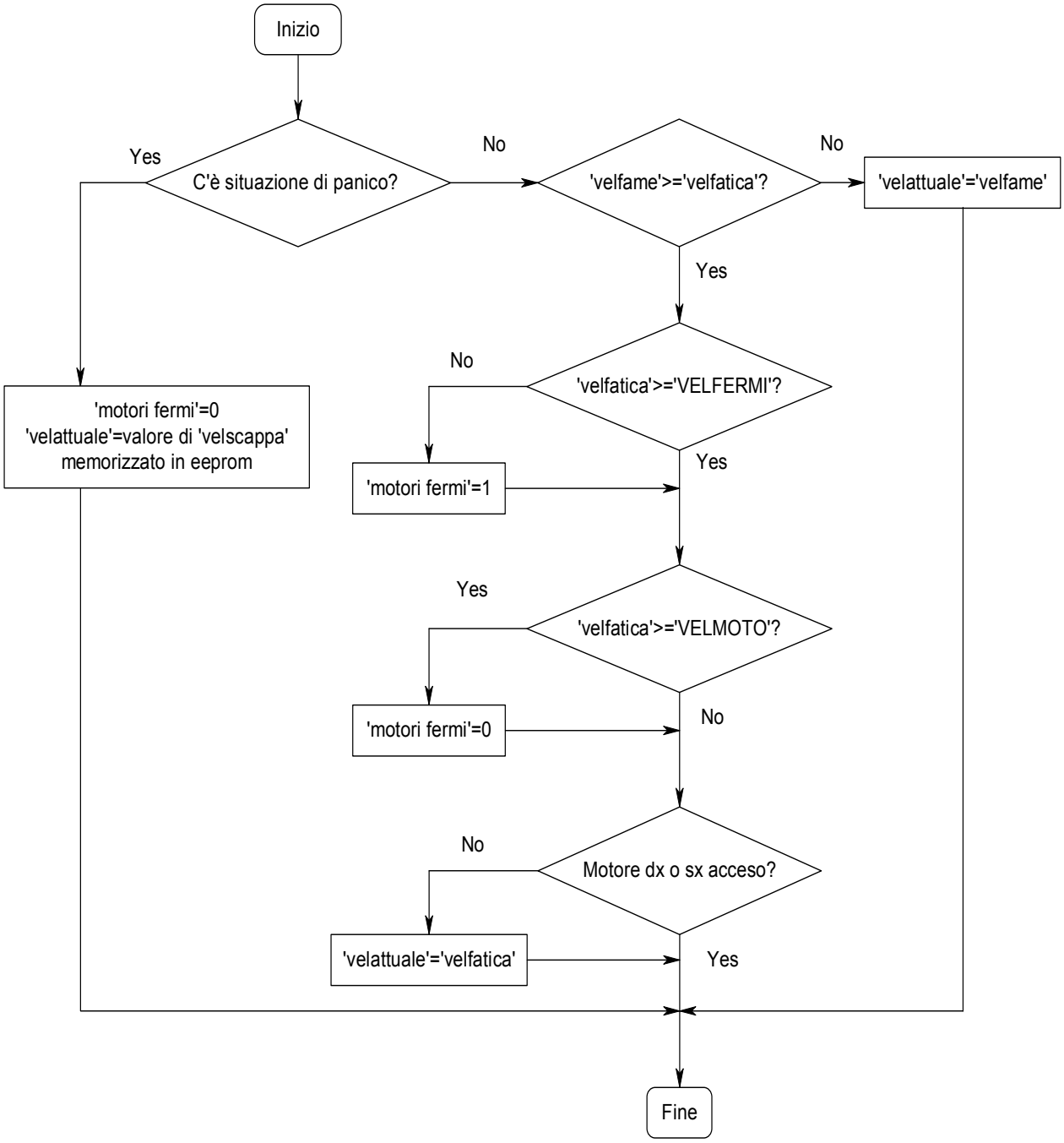

Figura 8 Diagramma di flusso della procedura 'velocitam'

La lettura della batteria prima veniva effettuata ogni 'intTimer1' interrompendo la corrente lettura dei sensori. Ora la batteria viene letta come un 9° sensore.

Procedure coinvolte: 'conversione'.

3.7. Gestione situazioni critiche

Per ottenere la sensibilità laterale richiesta abbiamo agito sulla interconnessione tra i blocchi dell'architettura di controllo.

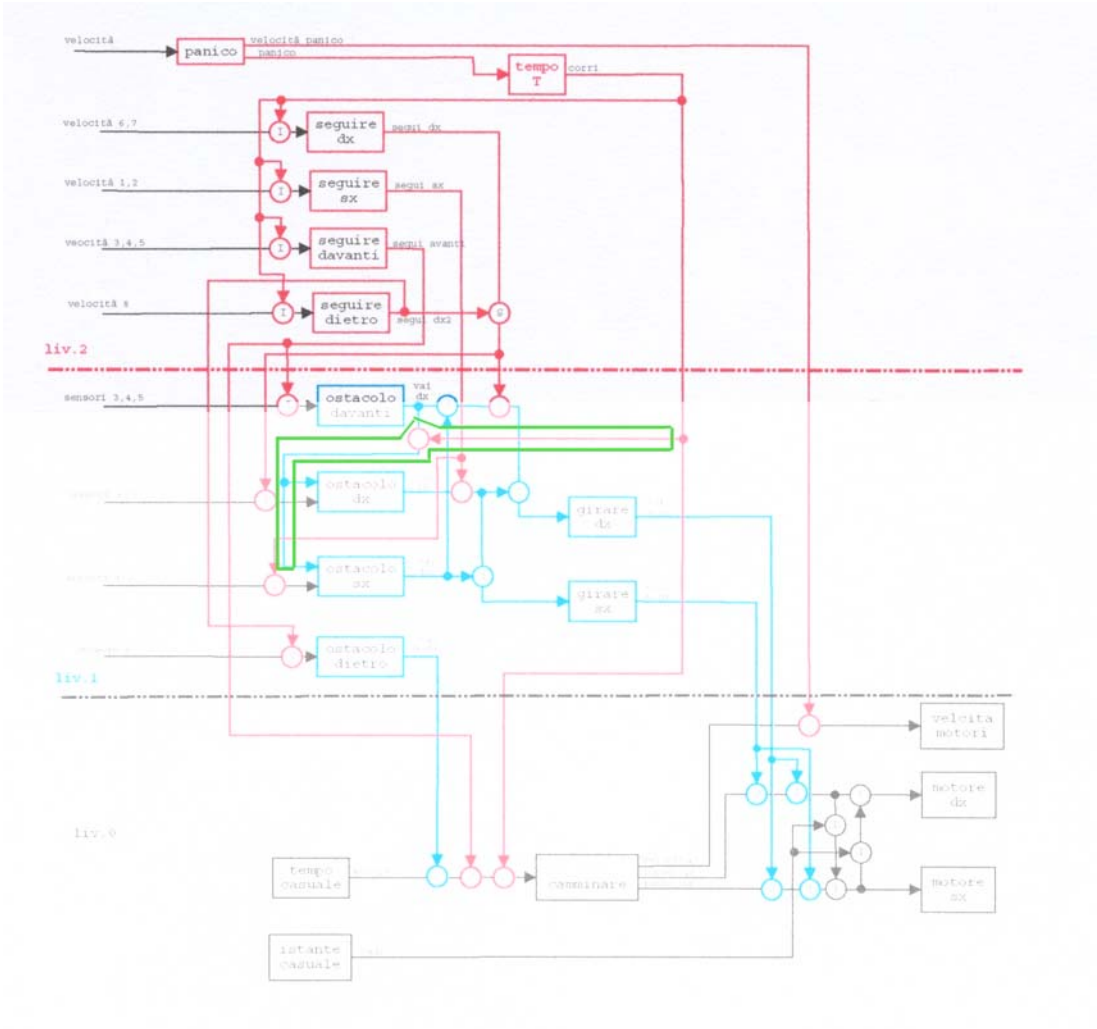

Figura 9 Schema dei primi tre livelli dell'architettura di controllo

In particolare come evidenziato nello schema sopra sono stati rimossi i collegamenti tra l'uscita del modulo 'ostacolo davanti' e gli ingressi di 'ostacolo dx' ed 'ostacolo sx'. Così facendo per la percezione degli ostacoli laterali non è più necessaria la presenza di un ostacolo davanti. Dal punto di vista del software eliminare le connessioni ha portato all'eliminazione dei controlli sulla variabile 'avanti' dalle procedure 'ostdestra' e 'ostsinistra'.

Procedure coinvolte: 'ostsinistra', 'ostdestra'.

Per rendere sensibile la sedia ad ostacoli molto vicini sono state apportate delle modifiche alla procedura 'sit_critica'. La nuova versione individua tre tipi di situazioni.

1. Un ostacolo posto lateralmente o davanti ad una distanza inferiore a 'CONTATTO' causa il blocco dei motori e l'accensione del buzzer. L'uscita da questo stato si ottiene rimuovendo l'ostacolo cioè quando i sensori rilevano che tutti gli ostacoli laterali e frontali sono a distanza superiore a quella indicata nella costante 'CONTATTO'.
2. Un ostacolo posto dietro ad una distanza inferiore a 'CONTATTO' causa l'accensione del buzzer e la commutazione nello stato di panico.
3. Ostacoli contemporaneamente presenti nelle direzioni avanti, destra e sinistra posti ad una distanza compresa tra 'CONTATTO' e 'VICINANZA' causano lo spegnimento dei motori e l'accensione del buzzer. L'uscita da questo stato si ottiene allontanando l'ostacolo davanti oltre i 50 cm e spostando oltre 'VICINANZA' o l'ostacolo sinistro o quello destro.

Procedure coinvolte: 'sit_critica'.

3.8. Collegamento seriale (RS232)

Sulla linea seriale nella fase di configurazione del PIC vengono inviati messaggi diagnostici che permettono di capire se un'operazione è stata completata con successo. Nel caso in cui essi risultino superflui possono essere saltati definendo diag=0.

La modalità di utilizzo della seriale è la seguente: avviato il programma, all'interno del ciclo infinito che continua a essere ripetuto in attesa di interruzioni, si controlla se è stato ricevuto un carattere proveniente dal terminale. A questo punto se il carattere premuto corrisponde allo spazio si entra in 'interact' dove vengono disabilitati gli interrupt globali, spenti i motori e si procede a una gestione a polling del collegamento seriale continuando a vedere se si ricevono ulteriori caratteri.

Non essendo abilitati gli interrupt l'unico modo per uscire da 'interact' è quello di digitare il carattere "X" e il programma viene riavviato.

Scegliendo la modalità di 'debugger' vengono visualizzate sul terminale i valori di alcune variabili utili a capire il corretto funzionamento della sedia.

Entrando in modalità 'memory' è possibile visualizzare e modificare il valore delle costanti memorizzate in eeprom. In particolare è implementato un ciclo con cui si possono visualizzare a video i valori delle prime 10 celle di eeprom essendo le uniche da noi utilizzate nel programma. Se viene digitato un nuovo valore diverso da quello attualmente in eeprom esso viene memorizzato sovrascrivendone il contenuto. Se il valore nuovo corrisponde a quello già memorizzato il programma evita di scrivere in eeprom per non rovinarla.

Nella modalità 'memoryrest' dopo richiesta di apposita conferma vengono riscritti in eeprom i valori predefiniti in fase di programmazione per le costanti. Per rendere più comprensibile il funzionamento della procedura 'car2num', essa è stata modificata e commentata abbondantemente.

4. Modalità operative

Per vedere il corretto funzionamento del programma è necessario avviare la sedia all'interno di un ambiente prestando particolarmente attenzione che non vi siano sul terreno oggetti sui quali la sedia salendoci si potrebbe ribaltare.

All'accensione si noterà che dopo aver riportato le gambe nella posizione di riposo la gamba sinistra tasterà il terreno per trovare la minima velocità di funzionamento, operazione che richiede dai 5 ai 15 secondi. Terminato questo periodo la sedia comincerà a muoversi in base agli stimoli esterni ed interni con le modalità descritte nel capitolo precedente. È necessario fare alcune precisazioni: durante le ultime prove effettuate si è visto come la sedia mentre cammina improvvisamente rileva un ostacolo inesistente molto vicino (suona il buzzer), va in panico e continua a rilevare per intervalli di tempo molto brevi oggetti vicini che in realtà non ci sono. Inoltre se fatta funzionare per un po' di tempo (circa 20 minuti) si può verificare una situazione in cui la sedia continua a muovere la stessa gamba risultando insensibile a qualunque ostacolo sia vicino sia lontano. Sollevandola abbiamo visto che quest'ultimo comportamento cessava immediatamente e la sedia ritornava al funzionamento corretto. Utilizzando il simulatore di sensori in sostituzione di quelli realmente montati sulla sedia non si sono mai verificati questi problemi confermando la nostra convinzione sul corretto funzionamento del programma. Alzando la sedia il parametro che cambia maggiormente è il carico sui motori e proporzionalmente la corrente da essi assorbita; tutto questo ci fa pensare che i problemi nascano dalla presenza di circuiti di potenza e di segnale sulla stessa scheda e da interferenze con i sensori. Consigliamo quindi per controllare il programma di usare la sedia con collegato il simulatore di sensori.

4.1. Uso della seriale

Una volta avviato un programma di emulazione di terminale, all'accensione della sedia compaiono a video i valori di 'velminima', 'velpaura', 'velscappa', 'velfermi', 'troppafame', 'duratapanico' seguiti dai messaggi diagnostici che indicano l'avvenuta esecuzione delle procedure di inizializzazione del PIC ('Diagnostic a', 'Diagnostic b', ..., 'Diagnostic i'). Da questo istante è possibile entrare in modalità 'interact' premendo il carattere 'spazio'. Consigliamo di premere il tasto 'Caps Lock' in quando ogni dato o comando inserito risulta interpretato correttamente dal programma solo se maiuscolo. In 'interact' il prompt ci indica che il programma sta aspettando la pressione di determinati tasti. Se viene digitato:

- "D" si entra in modalità 'debugger': vengono visualizzate alcune delle variabili di interesse per verificare lo stato di funzionamento. Per facilitare l'utente prima dei valori esadecimali appare sulla stessa riga una stringa che specifica il nome della variabile a cui si riferiscono. In ordine di visualizzazione avremo 'Batteria', 'Contafatica', 'Velfame', 'Velfatica', 'Velattuale', sensori da 1 a 8, 'Direzione'. Il programma aspetta poi la pressione del carattere "X" per tornare così al ciclo 'interloop'.

- "M" si entra in modalità 'memory alter mode': appare subito al terminale la frase ingannevole "Premere INVIO per continuare,X per uscire" che induce l'utente a premere il tasto "INVIO". In realtà è necessario digitare prima di tutto i due caratteri

relativi alla prima locazione di memoria (00,01,...,09) di cui interessa il contenuto seguiti dal carattere ":". Così facendo il programma visualizza in modo automatico a fianco dei ":" il valore attuale contenuto nell'indirizzo di eeprom digitato. Se qui si preme una "X" il programma torna al ciclo 'interloop', se si preme "INVIO" in modo automatico sulla riga successiva compaiono l'indirizzo della cella di eeprom successiva, i ":" e il suo contenuto. Continuando a premere "INVIO" è possibile visualizzare il contenuto di tutte le celle di eeprom usate dal programma e dopo la cella 09 si ritorna a visualizzare la cella 00. Se a seguito di una visualizzazione viene digitato un carattere diverso da "X" e "INVIO" esso viene interpretato come parte alta del valore in esadecimale che si vuol sostituire a quello attualmente presente in eeprom. Il programma attende in questo caso l'inserimento del carattere corrispondente alla parte bassa seguito dalla pressione del tasto "INVIO". Se il nuovo valore inserito è diverso da quello presente in eeprom, avviene la scrittura della memoria. In qualunque caso nella riga successiva viene ristampato l'indirizzo della cella coinvolta nell'operazione seguito dal valore ora presente in essa.

Esempio

In rosso le parti stampate dal programma, in nero le parti digitate a tastiera.

00:43<INVIO>

01:0A<INVIO>

02:B3A4<INVIO>

02:A4<INVIO>

03:12X

- "R" si entra in modalità 'Restoring EEPROM memory': compare subito la stringa "Ripristinare costanti predefinite?S=yes". Se si preme un carattere diverso da "S" si torna al ciclo 'interloop' altrimenti il programma reinizializza la eeprom ai valori imposti in fase di programmazione e torna automaticamente a 'interloop'.

- "X" si esce dalla modalità 'interact' e il programma viene riavviato da 'start'.

- un carattere diverso da quelli sopra elencati compare a video il messaggio "Unknow command" e si torna ad 'interloop' aspettando l'inserimento di un carattere valido.

4.2. Modalità di taratura

La sedia comprende un'autotaratura rispetto al terreno su cui è posta e quindi le procedure riguardanti i settaggi delle velocità non devono essere modificate in alcun modo.

4.3. Avvertenze

L'utilizzo della connessione seriale necessità di una particolare attenzione. Infatti tutti i caratteri tranne "INVIO" sono interpretati come semplici codici ascii e quindi ad esempio non è possibile cancellare un carattere digitato tramite la pressione di "backspace". Inoltre quando viene digitato un nuovo valore per la eeprom è necessario premere "INVIO" ed un eventuale errore di digitazione può essere corretto solo andando a riscrivere nuovamente nella eeprom.

Esempio

00:09a0<INVIO> (qui c'è stato l'errore di inserire "A" in minuscolo)

00:50A0<INVIO>

00:A0

Il funzionamento del programma ed in particolare dello sbilanciamento che talvolta affligge la sedia dipende in modo marcato dalla posizione delle masse (batteria, piombi sulle gambe posteriori, motori,...) sulla struttura. Modifiche sulla sedia rispetto alla versione lasciata all'artista Laura possono portare a perdere il gradito effetto di sbilanciamento ed anche in modo molto più grave ad impedire la progressione della sedia nonostante i passi si alternino correttamente. Ci sembra quindi importante ricordare la configurazione della sedia da noi adottata: assenza di piombi sulle gambe posteriori, batteria nella posizione originale, a riposo manovella del motore formante con il piano della sedia un angolo di circa 60° verso il basso.

5. Conclusioni e sviluppi futuri

Sono state realizzate le modifiche richieste per far sì che il programma risultasse funzionante sul prototipo e garantisse un funzionamento il più prossimo possibile a quello di una persona mutilata cercando di mantenere la struttura portante del software invariata. La gestione della seriale risulta ora funzionale alla messa a punto del software. Possibili sviluppi futuri possono essere:

- inserimento di un bootloader che facilita la programmazione del PIC;
- separazione tra circuiti di potenza e di segnale per evitare i malfunzionamenti evidenziati;
- minimizzazione delle interferenze sui sensori.

Bibliografia

La parte di documentazione disponibile in internet è seguita dall'indirizzo dove può essere reperita.

[1] Microchip: “PIC16F87X Data Sheet 28/40-Pin 8-Bit CMOS FLASH Microcontrollers”, www.microchip.com .

[2] Microchip: “PICmicro™ Mid-Range MCU Family Reference Manual”, www.microchip.com

[3] Sharp: ”GP2Y0A02YK Long Distance Measuring Sensor”, www.sharpsma.com

[4] Zanacchi, A.: "Studio di una 'subsumption architecture' evoluta per l'emulazione di comportamenti umani", Tesi di laurea, Brescia, giugno 2003.

[5] Microchip: "AN566 Using the PORTB Interrupt on Change as an External Interrupt", www.microchip.com

[6] Microchip: "AN774 Asynchronous Communications with the PICmicro USART", www.microchip.com

Indice