POLITECNICO DI MILANO
Polo Regionale di Como

Facoltà di Ingegneria dell'Informazione
Corso di studi in Ingegneria Informatica

CONTROLLO DI UNA CARROZZINA CON BIOSENSORI A BASSO COSTO
Laboratorio di Robotica e Intelligenza Artificiale
del Politecnico di Milano

Relatore: Prof. Andrea BONARNI

Tesi di Laurea di:
Roberto VANDONE matr. 669565

Anno Accademico 2008-2009

Sommario

Le “Neural Interface” (NI) o “Brain-Computer Interface” (BCI) sono dei dispositivi che vanno posizionati sulla testa e che leggono i segnali elettromagnetici che provengono dal cervello. Tramite appositi software è possibile decodificare i segnali letti e permettere con questi il pilotaggio di svariati dispositivi. Lo scopo della tesi è quello di utilizzare un biosensore a basso costo per poter comandare i movimenti di una carrozzina automatizzata. Per raggiungere questo obiettivo ci si è affidati ad un dispositivo commerciale a basso costo, il NIA (Neural Impulse Actuator) della OCZ, che ci ha permesso di movimentare la carrozzina Lurch.

Ringraziamenti

Ringrazio innanzitutto il Prof. Andrea Bonarini per avermi seguito durante la stesura della tesi ed avermi dato la possibilità di lavorare a questo progetto.

Vorrei anche ringraziare tutti i professori della IOL per avermi dato l'opportunità di realizzare il sogno della laurea.

In ultimo, ma non meno importanti, ringrazio la mia famiglia, Chiara e Francesco, per avermi supportato e sopportato durante questi anni di studio.

Indice

1 Introduzione

In questi ultimi anni si stanno sviluppando le cosiddette interfacce neurali, dispositivi in grado di leggere i segnali elettromagnetici provenienti dal cervello.

Queste interfacce rappresentano un mezzo di comunicazione diretto tra cervello e un dispositivo esterno come ad esempio un computer.

Il nome di questi dispositivi è *Brain-Computer Interface*, abbreviato in BCI, i quali ricevono i comandi direttamente da segnali derivanti da attività celebrare, come ad esempio il segnale elettroencefalografico.

Ne esistono principalmente tre categorie:

- Le BCI invasive che consistono nell'impiantare un dispositivo direttamente nella materia grigia e che permettono di acquisire segnali di altissima qualità, ma oltre ai rischi connessi ad un'operazione di neurochirurgia, vi è il problema dei tessuti cicatrizzanti, che causano un progressivo indebolimento del segnale fino alla sua perdita totale. Vi è anche il problema della possibile reazione dell'organismo ad un corpo estraneo all'interno del cervello.
- Le BCI parzialmente invasive che consistono nell'impiantare il dispositivo all'interno del cranio, ma fuori dal cervello. I segnali hanno una qualità minore rispetto alla categoria precedente ma hanno un minor rischio di formazione di tessuto cicatrizzante.
- Infine esistono le BCI non invasive che consistono in sensori "indossabili" i quali ovviamente sono molto più comodi ed utilizzabili rispetto ai precedenti, ma che producono un segnale molto povero in

quanto il cranio attenua le onde elettromagnetiche create dai neuroni, ed inoltre diventa più difficile determinare l'area del cervello che li ha creati.

Il tipico campo di applicazione è quello di supporto funzionale ed ausilio per persone con disabilità, anche se negli ultimi due anni sta cercando di affermarsi anche come interfaccia ludica per il controllo di videogiochi.

Nel campo medico sono già presenti diverse applicazioni rivolte appunto a persone con disabilità motorie, ci sono degli studi rivolti alla possibilità di utilizzare le BCI per la sintesi vocale di un set predefinito di parole, per la scrittura o la navigazione sul web o ancora per il controllo di sedie a rotelle su percorsi predefiniti.

Lo scopo di questa tesi è quello di pilotare una sedia a rotelle tramite una di queste interfacce già presenti sul mercato consumer e con un costo molto limitato.

Ovviamente si tratta di un'interfaccia non invasiva che, tramite una fascia posta sulla fronte, rileva tre tipologie di informazioni, l'attività muscolare del viso (elettromiogramma), i movimenti degli occhi (elettrooculogramma) e l'attività del cervello (elettroencefalogramma).

Trattandosi di un dispositivo a basso costo, il prezzo è inferiore ai 100,00 €, il segnale relativo all'attività cerebrale non è risultato facilmente utilizzabile, così come il segnale relativo ai movimenti oculari non sono di facile utilizzo. Ci si è quindi concentrati sui segnali provenienti dall'attività muscolare per poter comandare la carrozzina.

Grazie poi al software già sviluppato sulla carrozzina Lurch, che include diversi automatismi alla guida assistita, è stato possibile raggiungere lo scopo preposto in modo molto proficuo.

Come accennato precedentemente, sono già state fatte delle ricerche e degli

sviluppi per il controllo tramite interfacce neurali di sedia a rotelle per disabili. Lo stesso progetto Lurch, del laboratorio di intelligenza artificiale e robotica del Politecnico di Milano, ha sviluppato un sistema di interfacciamento alla carrozzina, che permette, tramite la sola attività cerebrale, di poter selezionare una meta, fra quelle disponibili, dove la carrozzina si dovrà dirigere. In questo caso si è utilizzato un elettroencefalografo, come quelli presenti nelle normali aziende ospedaliere, quindi con un costo non indifferente, con il quale si è stati in grado di rilevare quella che viene chiamata onda P300, che viene generata quando la persona vede un immagine che stava aspettando. In questo caso, vengono trasmessi ad un monitor collegato alla carrozzina i nomi delle possibili destinazioni che sono state programmate sulla stessa, e quando l'utente riconosce il nome che stava aspettando, viene generata dal suo cervello questa onda, che viene immediatamente riconosciuta dal sistema ed utilizzata come evento per poter comandare alla carrozzina di raggiungere la destinazione desiderata.

Questo sistema ha ovviamente dei pregi e dei difetti. Iniziando dai pregi, possiamo sicuramente affermare che il sistema è molto affidabile e che è utilizzabile anche su pazienti completamente paralizzati, ma di contro è sicuramente molto costoso e poco pratico, dovendo collocare tutti gli elettrodi necessari all'elettroencefalografo previa applicazione dell'apposito gel. Inoltre si può affermare che la maggior parte dei pazienti che possono necessitare di una carrozzina di questo tipo sono solo parzialmente paralizzati, in pochi casi la paralisi colpisce tutto il volto.

Il sistema che viene proposto con questa tesi permette di coprire una buona parte delle disabilità più comuni utilizzando un dispositivo a basso costo e molto pratico da utilizzare. Certamente, ci sono anche in questo caso delle limitazioni dovute in primo luogo alla giovinezza della tecnologia e in secondo luogo

dell'economicità dei componenti utilizzati per poter tenere il prezzo del dispositivo così basso. Ad esempio l'hardware risente fortemente dei disturbi elettromagnetici presenti nell'ambiente di utilizzo. Inoltre, come verrà approfondito meglio in seguito, il dispositivo è stato pensato per un utilizzo domestico e quindi ci sono delle limitazioni sia nel software fornito con il dispositivo che non include (almeno per il momento) alcun SDK per il suo interfacciamento, sia limitazioni ai sistemi operativi sui quali si può utilizzare, al momento, solo Windows.

La tesi è strutturata come segue:

- Nella seconda sezione vengono descritti i principi di funzionamento di queste interfacce, presi in considerazione i dispositivi attualmente in commercio e i motivi che hanno portato alla scelta del NIA.
- Nella terza sezione vengono descritte in modo dettagliato le prove che sono state effettuate sul dispositivo, limiti e prospettive.
- Nella quarta sezione viene descritta l'interfaccia utente adottata e le modalità di interazione con l'utente, nonché l'architettura del sistema sviluppato.
- Nella quinta sezione verrà tracciata una possibile espansione per il futuro.
- Nell'appendice A è riportato il manuale d'uso del programma sviluppato.
- Nell'appendice B sono riportati i sorgenti del programma sviluppato.

2 Stato dell'arte

Attualmente ci sono diverse università, ricercatori, laboratori e società che stanno studiando nuove tecnologie e sperimentando le applicazioni di queste interfacce neuronali.

2.1 Principi di funzionamento

Il funzionamento di questi dispositivi si basa in buona parte sulla misurazione di segnali elettrici che vengono generati dal nostro sistema nervoso. Vengono ora analizzate le tecniche e i principi sui quali si basano gli studi sulle interfacce neurali.

L'elettroencefalogramma (EEG) registra l'attività elettrica cerebrale tramite degli elettrodi di superficie posizionati sulla testa, un caschetto per l'elettroencefalogramma è visibile in Figura 2-1. Questa tecnica inventata nel 1929 da Hans Berger, consiste appunto nel misurare le piccole differenze di potenziale elettrico (microvolt) che vi sono tra i vari punti del cuoio capelluto, Figura 2-2. I potenziali che vengono misurati in superficie sono principalmente il risultato dell'attività dei neuroni corticali piramidali disposti in corrispondenza dell'area corticale sottostante l'elettrodo. Pertanto i potenziali misurati sono il frutto dell'attivazione di un gran numero di neuroni nell'area sottostanze all'elettrodo. E' quindi possibile conoscere in tempo reale le aree del cervello più attive in un determinato istante. Per esempio, per attivare il movimento di un muscolo, si attivano diversi neuroni nel nostro cervello, quindi tramite EEG è possibile registrare questa azione. Anche l'immaginare di compiere un'azione provoca l'attivazione di diversi neuroni e quindi la possibilità di rilevare questa intenzione. Utilizzando questi meccanismi è possibile ricavare dei segnali di controllo da utilizzare per i

sistemi BCI.

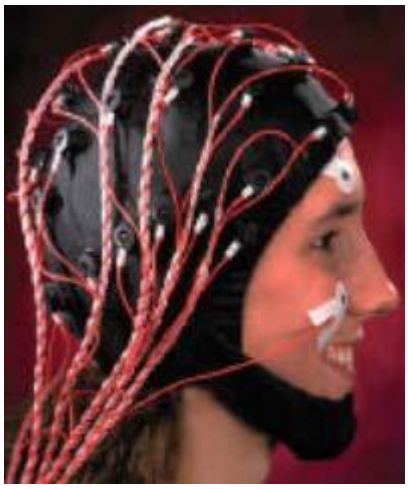

Figura 2-1 Caschetto per elettroencefalogramma.

Figura 2-2 Esempio di tracciato elettroencefalografico.

Ad oggi si possono classificare diversi segnali di controllo. Innanzitutto bisogna definire due categorie:

- i segnali BCI dipendenti (o secondari). In questa categoria appartengono i segnali che potrebbero esseri rilevati "monitorando" le

normali vie d'uscita del cervello. Un esempio sono i segnali relativi all'attività muscolare degli occhi per determinare la direzione dello sguardo. Si tratta quindi di rilevare tramite EEG la direzione dello sguardo piuttosto che monitorare la posizione degli occhi.

Un BCI dipendente è essenzialmente un metodo alternativo per il rilevamento di messaggi per vie canoniche e, nonostante non fornisca un nuovo canale di comunicazione che sia indipendente da quelli convenzionali, può essere utile.

- i segnali BCI indipendenti (o primari), che sono tutti quei segnali che non sono trasportati da nervi e muscoli periferici.

Tra i segnali indipendenti possiamo citare il metodo del potenziale evocato P300 (accennato in precedenza) e i ritmi Sensorimotori (SMR) utilizzati dal NIA.

Il P300 o "oddball" consiste nel rilevare nell'EEG un picco dopo circa 300 msec dall'applicazione al soggetto di uno stimolo che può essere uditivo visivo o somatosensoriale e del quale il soggetto è in attesa. Una tipica applicazione è quella di visualizzare una tastiera virtuale sullo schermo con le lettere che si illuminano in sequenza, quando la lettera che il paziente sta aspettando si illumina, si avrà il picco dopo 300 msec, il picco risulta essere misurato sopra la corteccia parietale.

La misurazione dei ritmi Sensorimotori consiste nel monitorare le attività cerebrali delle frequenze comprese tra gli 8 e i 12 Hz. Queste attività, quando sono misurate sopra la corteccia visiva, prendono il nome di ritmo alfa. Queste onde sono direttamente legate ai canali motori naturali del cervello. Ad esempio, un movimento o la sua preparazione, è tipicamente accompagnato da un decremento nell'attività delle onde alfa e beta (18-26Hz). Ma non solo, queste onde variano anche con la sola immaginazione motoria e quindi si prestano molto bene come segnali di controllo.

E' quindi possibile imparare a controllare l'ampiezza di queste onde

sfruttandole come segnali di controllo per l'attivazione dei dispositivi esterni.

Rispetto al metodo precedente, che non richiedeva alcun addestramento da parte del paziente, questo metodo deve essere "imparato" e i tempi d'apprendimento non sono rapidissimi. Si deve iniziare imparando a rilassare il corpo, poi ad immaginare di muovere una mano, poi tutto il corpo ecc., tutto questo per poter effettuare il controllo. Andando però avanti con l'addestramento ci si accorge che l'immaginazione diventa meno importante e il controllo diventa più naturale senza dover pensare ai dettagli della prestazione.

L'elettromiogramma (EMG) registra l'attività elettrica muscolare. Questo esame medico è molto utilizzato in neurologia o in ortopedia e ha come scopo l'analisi dell'attività muscolare. L'esame vero e proprio è un esame invasivo, in quanto vengono applicati dei piccoli aghi nel muscolo sui quali vengono misurati i potenziali durante le varie fasi di attività del muscolo, dal riposo alla contrazione massima. Esso rappresenta un metodo affidabile per dare informazioni sulla funzionalità dei nervi periferici e dei muscoli scheletrici.

Nelle interfacce di BCI in realtà le misurazioni vengono effettuate con dei semplici elettrodi superficiali, non essendo necessari tutti i dati rilevati nell'esame medico.

Infine l'elettrooculogramma (OEG) che come nel caso precedente, viene utilizzato in medicina con finalità e modalità molto più complesse di quelle necessarie per i nostri scopi. In particolare l'esame serve per verificare il funzionamento dell'epitelio pigmentato retinico (RPE), lo strato di cellule situato tra la retina e la coroide. Nelle interfacce BCI si sfrutta la differenza di potenziale esistente tra la cornea (positiva) e la retina (negativa), schematizzato in Figura 2-3, per individuare i movimenti dello stesso, questo grazie a 2 elettrodi posizionati in prossimità dell'occhio, come visibile in Figura 2-4.

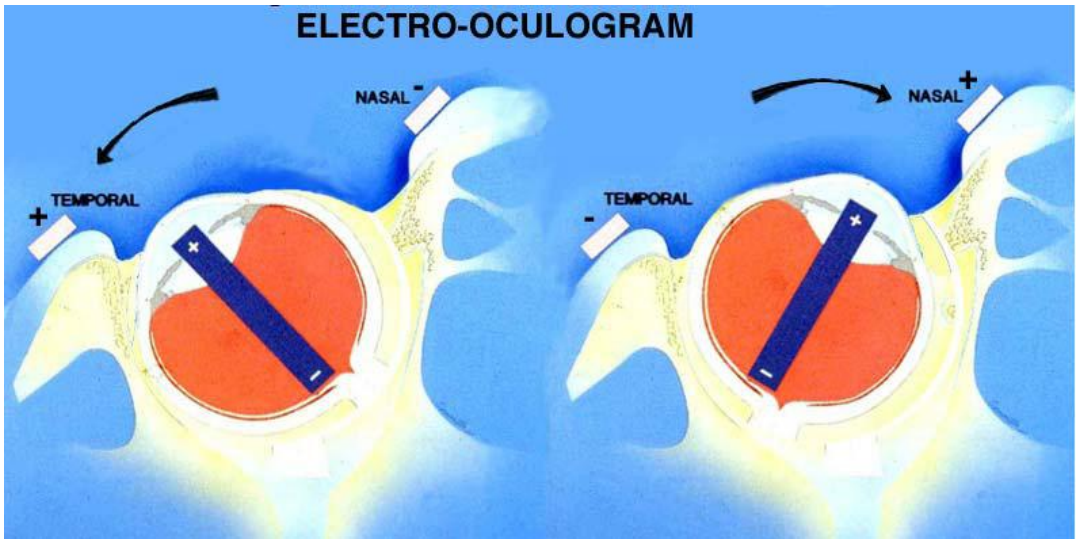

Figura 2-3 Schema elettrico degli occhi (bipolo oculare).

Figura 2-4 Applicazione medica dei sensori per l'OEG.

Oltre a queste tecniche già oggi utilizzate per le interfacce neurali, vi sono allo studio altri sistemi basati su tecniche derivate da altrettante applicazioni mediche, come ad esempio la Magnetoencefalografia (MEG), che misura i campi magnetici generati dall'attività cerebrale, e la Functional Magnetic Resonance Imaging (FMRI) che misura l'afflusso di sangue ossigenato al cervello durante la sua attività.

2.2 Dispositivi commerciali

Come si è appena visto, tutte queste tecniche richiedono strumentazione e personale specializzato con dei costi molto elevati. Stanno però nascendo dei dispositivi che utilizzano i principi visti poc'anzi ma con costi decisamente più bassi tuttavia non raggiungendo lo stesso risultato, sia in termini di affidabilità che di qualità delle informazioni. Questi dispositivi commerciali sono già disponibili sul mercato e possono essere utilizzati per sviluppi ad hoc o per il controllo di videogiochi.

I tre prodotti principali sono:

Il Neural Impulse Actuator (NIA) della OCZ visibile in Figura 2-5, è composto da una fascetta con tre sensori che permette di rilevare le attività muscolari, le attività del cervello e i movimenti dell'occhio. Questo dispositivo viene venduto principalmente per scopi ludici, comprende un software che permette di creare dei profili personalizzati per poter appunto giocare. Al momento non è disponibile un SDK per fare degli sviluppi in proprio e l'unico sistema operativo supportato è Windows (requisito minimo XP). Il costo è di circa 100,00 €.

Figura 2-5 Il NIA della OCZ
(http://www.ocztechnology.com/products/ocz_peripherals/nia-neural_impulse_actuator).

L'EPOC della Emotiv, visibile in Figura 2-5, questo prodotto si basa su ben 14 sensori e un giroscopio. Utilizza la tecnologia wireless per comunicare con il computer, questa interfaccia sembra essere in grado di rilevare le emozioni dell'utente quali ad esempio eccitazione, noia, frustrazione, espressioni facciali, e altre attività cerebrali, oltre alle rotazioni della testa (tramite il giroscopio). In questo caso vi sono diverse versioni del prodotto, alcune delle quali includono anche un SDK per poter fare degli sviluppi personalizzati sull'oggetto. Come per il NIA l'unico sistema operativo supportato risulta essere Windows.

Il costo in questo caso varia dai $500,00 ai $7.500,00 in base alla versione e alla presenza o meno di tool di supporto alla programmazione (SDK).

Figura 2-6 EPOC della Emotive (http://www.emotiv.com/).

Infine l'ultimo prodotto commerciale disponibile sul mercato è il Mindset della NeuroSky, Figura 2-7, questo dispositivo dispone di un solo sensore per l'EEG, è anch'esso wireless ed è compatibile sia con Windows che con MAC. In questo caso viene venduto con un completo SDK. Il suo costo è di circa 150,00€.

Figura 2-7 Mindset della NeuroSky (http://www.neurosky.com/).

Come si può vedere dalle immagini tutti questi prodotti commerciali sono facilmente indossabili e non richiedono particolari attenzioni per poter essere utilizzati.

In commercio esistono poi dei giochi che si basano principalmente sui prodotti appena descritti come ad esempio il Mind-Flex (http://mindflexgames.com/) della Mattel oppure il MindBall (http://www.mindball.se/product.html).

2.3 Scelta del dispositivo

Il più completo e affidabile dei tre dispositivi attualmente in vendita risulta essere l'EPOC il quale ha le maggiori possibilità di utilizzo, ma purtroppo risulta anche essere il più caro fra tutti. Inoltre la sua disponibilità sul mercato è molto recente (dicembre 2009) e non sono presenti ancora recensioni di utenti che hanno utilizzato il prodotto. Di contro il Mindset già presente sul mercato da metà 2009, risulta troppo limitato per la presenza di un solo sensore e per la possibilità di utilizzare solo l'attività celebrare per il controllo dei dispositivi. Almeno fino a poco tempo fa era venduto solo alle aziende e non a privati e quindi anche in questo caso non sono presenti feed-back da parte degli utenti.

La scelta a questo punto risulta quasi obbligata verso il NIA che è già presente sul mercato da aprile/maggio 2008.

Come detto in precedenza al momento non è disponibile alcun SDK per poter fare degli sviluppi mirati sulle funzionalità del dispositivo. Esistono però in rete diversi progetti open source che si stanno muovendo per potersi interfacciare direttamente con il dispositivo. Tutti questi progetti al momento si stanno concentrando solo sulla decodifica dei tracciati EEG rilevati dal dispositivo, e non sulla decodifica dei movimenti muscolari o di quelli oculari. Tutto questo interesse

dimostrato in rete per questo dispositivo ci ha portati ad acquistarlo per poter vedere se utilizzabile non solo per scopi ludici, ma anche per un utilizzo con persone disabili per il controllo appunto di una carrozzina. A questo proposito, sempre in rete, vi sono riferimenti ad altri progetti in corso di sviluppo, che utilizzano il NIA come aiuto alle persone disabili. Un esempio di questo è il progetto Cesar-X (http://nuke.amicidieluana.it/ProgettoELU1/ProgettoCesareX/tabid/507/Default.aspx), che si prefigge lo scopo di utilizzare il NIA per poter permettere a persone con disabilità molto gravi o gravissime di poter comunicare dandogli la possibilità di scrivere al computer.

3 Prove con il NIA

L'idea iniziale era quella di utilizzare solo le onde Alfa e Beta, che sono le onde cerebrali che caratterizzano gli stati di rilassamento e meditazione (alfa) e di veglia (beta), rilevate dal NIA e di conseguenza solo i segnali derivanti dall'attività cerebrale (EEG). L'idea era quindi quella di rendere il prodotto utilizzabile anche da persone con paralisi completa. Si è infatti proceduto a sviluppare il software necessario all'acquisizione dei dati direttamente dal dispositivo hardware, seguendo le indicazioni riportate sui vari forum preposti a trattare questo dispositivo. Il software, inizialmente sviluppato su Windows, si occupava di acquisire i dati dalla porta USB (porta alla quale è connesso il NIA) e tramite elaborazione digitale del segnale in tempo reale (Filtri digitali per l'eliminazione del rumore e FFT per la separazione delle frequenze), risalire alle informazioni necessarie per l'interfacciamento richiesto. Il software poteva poi essere portato su altre piattaforme come ad esempio Linux.

Durante le prove ci si è però accorti che il NIA risultava essere troppo sensibile e che i segnali, specialmente quelli cerebrali, risultavano essere molto disturbati da campi elettromagnetici esterni, come ad esempio motori elettrici, trasformatori, router wireless o anche il solo PC se non ben schermato. Questo perché, come spiegato nel capitolo precedente, i segnali rilevati dagli elettrodi hanno un potenziale elettrico molto basso e quindi necessitano di una forte amplificazione. Di conseguenza, se le schermature dei collegamenti non sono più che perfette, si ha un'amplificazione anche dei più piccoli disturbi. Oltretutto la frequenza della corrente elettrica in Italia è di 50Hz che è molto vicina alle frequenze che devono essere elaborate dal sistema. Inoltre, anche lavorando in ambienti schermati, il

controllo di queste onde è risultato molto difficile e richiede diverso tempo per essere appreso, giusto per dare un idea si può vedere questo studio eseguito su un dispositivo molto simile al NIA (http://www.brainfingers.com/BATFinalNIH1.PDF). Purtroppo non sono solo i disturbi elettromagnetici a modificare i segnali, ma anche i movimenti oculari e muscolari hanno pesanti influenze sui segnali che quindi risultavano dipendenti anche da questi fattori. Questo è un problema già noto in letteratura e si stanno studiando degli algoritmi per poter eliminare questi "disturbi" relativi ai segnali EOG e EMG. Facendo delle prove ci si rende subito conto che per una persona non paralizzata il tentativo di controllare queste onde si traduce anche in piccoli movimenti muscolari che portano appunto ad un controllo anche su queste frequenze.

Infine si sono evidenziate delle piccole differenze rispetto all'elaborazione del segnale eseguito dal nostro software e l'elaborazione risultante dal software dato in dotazione con il NIA.

A questo punto, abbiamo contattato la OCZ per presentare il lavoro che si stava svolgendo in questa tesi e per chiedere una collaborazione con loro per poter ottenere dei risultati migliori. Durante questo scambio di corrispondenza elettronica ci è stato detto che purtroppo sia l'hardware che il software sono coperti da segreto industriale e che per questa ragione non era possibile fornirci alcuna informazione né sul protocollo di comunicazione con il NIA né tanto meno sugli algoritmi di decodifica dei segnali letti.

Siamo stati però messi in contatto con la persona che ha brevettato il dispositivo e sviluppato l'intero software il quale ci ha detto che stanno lavorando allo sviluppo di un SDK e, in via ufficiosa, ci è stata rilasciata una versione beta per poter continuare il progetto e per collaborare con loro nella fase di test del software.

L'unico ostacolo è che gli sviluppi saranno solo sulla piattaforma Windows ed in particolare utilizzano il framework .net.

Questa situazione è intesa come ostacolo in quanto tutto il software attualmente sviluppato sulla carrozzina Lurch è sotto Linux. Il dover utilizzare un secondo sistema operativo ci costringe a dover dotare la stessa di un nuovo computer per poter far funzionare il driver NIA.

Siamo stati inoltre informati dello sviluppo di una nuova versione del dispositivo che conterrà delle importanti novità. In primo luogo si sta lavorando per risolvere il problema delle interferenze che attualmente affligge questa prima versione, inoltre la prossima cuffietta sarà wireless così da poter essere utilizzata anche non a contatto con un computer. Infine avrà anch'essa un giroscopio per rilevare le rotazioni della testa. Al momento non è però ancora disponibile una data, neppure ufficiosa, di rilascio.

Viste quindi tutte le problematiche e le limitazioni che affliggono i segnali cerebrali, si è deciso di utilizzare i soli segnali muscolari, in quanto anche quelli oculari non sono semplicissimi da gestire e durante il movimento della carrozzina viene naturale guardarsi intorno, rischiando di innescare movimenti non desiderati. Questo ci ha permesso di avere un controllo molto più affidabile e un tempo di apprendimento da parte dell'utilizzatore molto più veloce. Inoltre, il numero delle persone con paralisi totale che utilizzano una sedia a rotelle è molto piccolo.

Di contro, c'è anche da segnalare che l'utilizzo intensivo della modalità di movimentazione libera della carrozzina porta a un certo affaticamento ai muscoli facciali utilizzati per il controllo, almeno nelle prime fasi di utilizzo. Probabilmente, come accade con tutti i muscoli, con l'allenamento e l'utilizzo costante questi fastidi tendono a sparire.

4 Interfaccia utente e architettura del sistema

Al fine di soddisfare il maggior numero di utenti, si è deciso di sviluppare due distinte modalità di funzionamento del programma di interfacciamento tra il NIA e la carrozzina.

Una utilizzabile da persone con disabilità maggiore, e che consente a queste persone di spostarsi a scelta tra una lista di luoghi precedentemente configurati sulla carrozzina, e una modalità che permette la movimentazione libera della stessa.

4.1 Modalità con destinazioni pre-impostate

Il software visualizza, all'interno del monitor, le immagini delle possibili destinazioni configurate vedi Figura 4-1, o in alternativa, se l'immagine non fosse disponibile, il nome della destinazione Figura 4-2. Queste immagini vengono visualizzate per un tempo configurabile in modo ciclico. Quando l'utente vede la destinazione nella quale si vuole recare non deve far altro che attivare i muscoli facciali, per esempio può muovere la fronte, oppure serrare le mascelle. Nel momento il cui il NIA rileva l'attivazione dei muscoli appare immediatamente una nuova finestra che chiede la conferma della destinazione selezionata. Questa finestra di conferma, visibile in Figura 4-3, ha lo scopo di filtrare eventuali selezioni non desiderate, dovute a movimenti non finalizzati allo spostamento. La finestra rimane visibile per un periodo di tempo anche esso configurabile; se l'utente non compie alcuna azione, allo scadere del tempo, la finestra scompare e la sequenza di

immagini riprende. Al contrario, se durante la visualizzazione della finestra l'utente muove nuovamente i muscoli facciali, il software invia il comando alla carrozzina che si sposterà alla destinazione selezionata.

Di seguito sono riportate alcune immagini relative a questo primo modo di utilizzo del sistema.

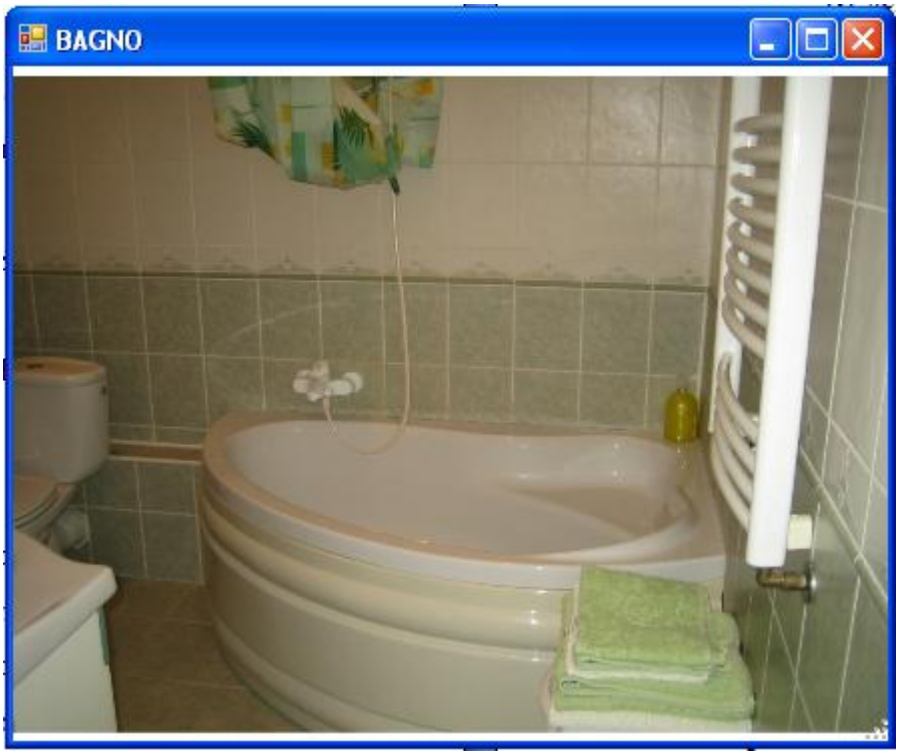

Figura 4-1 Visualizzazione di una possibile destinazione.

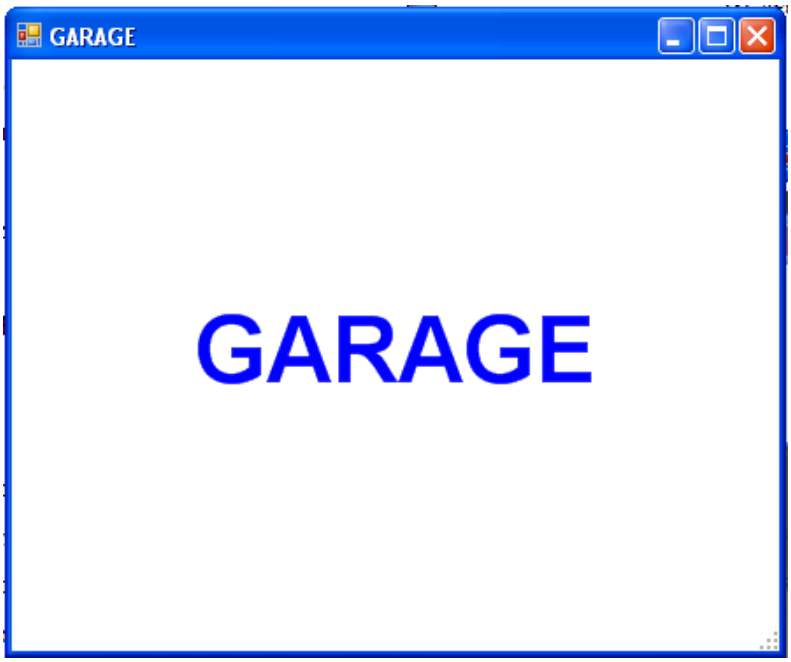

Figura 4-2 Visualizzazione di una destinazione senza immagine.

Figura 4-3 Finestra di conferma della destinazione, usata per evitare la selezione involontaria di una destinazione.

La dimensione e il formato delle immagini che sono associate alle singole destinazioni posso essere differenti. Sarà compito del software stesso occuparsi di caricare i vari formati supportati e di scalare l'immagine in modo tale che occupi il maggior spazio disponibile, mantenendo però costanti le proporzioni della stessa.

4.2 Modalità movimento libero

In questo caso il controllo da parte dell'utente risulta essere più impegnativo, ma di contro permette di potersi muovere liberamente anche in ambienti nuovi. Questo risultato è stato possibile anche grazie ai sensori presenti sulla carrozzina che evitano che manovre errate, dovute sia a errori da parte dell'utente che da eventuali errori da parte del NIA, scongiurino il pericolo di urto per l'utente stesso o per le persone/cose che gli stanno intorno.

Le azioni disponibili in questa modalità sono fondamentalmente quattro, la possibilità di far avanzare la carrozzina, la possibilità di farla retrocedere, la possibilità di farla ruotare su se stessa verso destra o sinistra.

Tramite un segnale muscolare di intensità non elevata si scorrono le quatto possibili azioni da eseguire (avanzamento, rotazione a destra, rotazione a sinistra e retromarcia), in successione.

Tramite un segnale muscolare di intensità elevata si attiva l'opzione selezionata, in caso di avanzamento o retromarcia, la carrozzina continua l'azione fino all'esecuzione di un nuovo comando di pari intensità, mentre nel caso di comando di rotazione, viene eseguita una rotazione tipicamente di 90° (parametro configurabile). Anche in questo caso se fosse necessario interrompere la rotazione prima che sia completa, basterà applicare un nuovo movimento muscolare di pari intensità che fermerà la rotazione.

4.3 Architettura

L'architettura visibile in Figura 4-4 è abbastanza semplice. Il software sviluppato in questa tesi riceve in ingresso dei segnali, in particolare il driver del NIA simula la pressione di alcuni tasti configurabili. È infatti in questo ambiente (il software fornito con il dispositivo) che avvengono le calibrazioni e le configurazioni necessarie al rilevamento dei biosegnali utilizzati dal software di controllo della carrozzina.

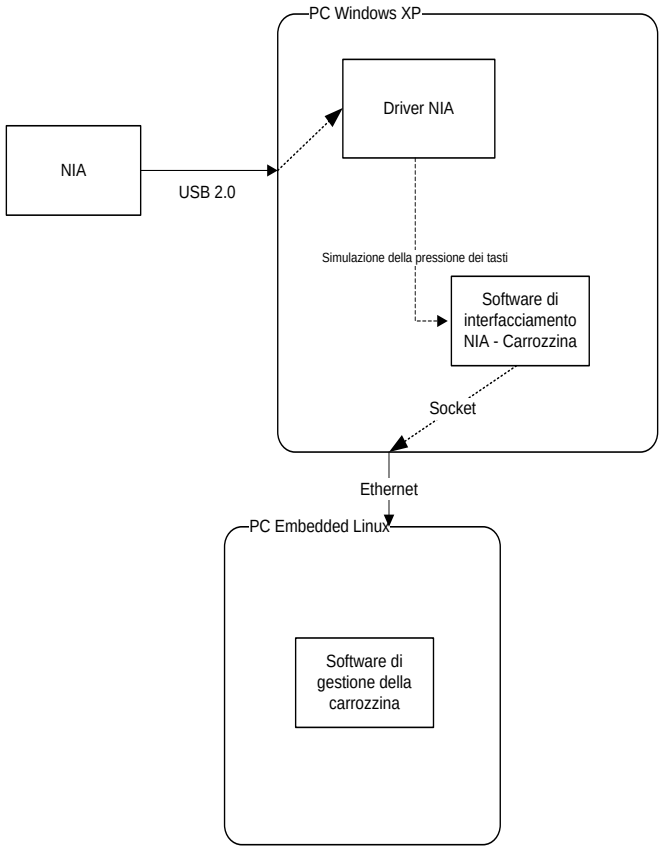

Figura 4-4 Architettura del sistema

Nella Figura 4-5 è riportata la schermata del NIA che visualizza le ampiezze dei segnali processati.

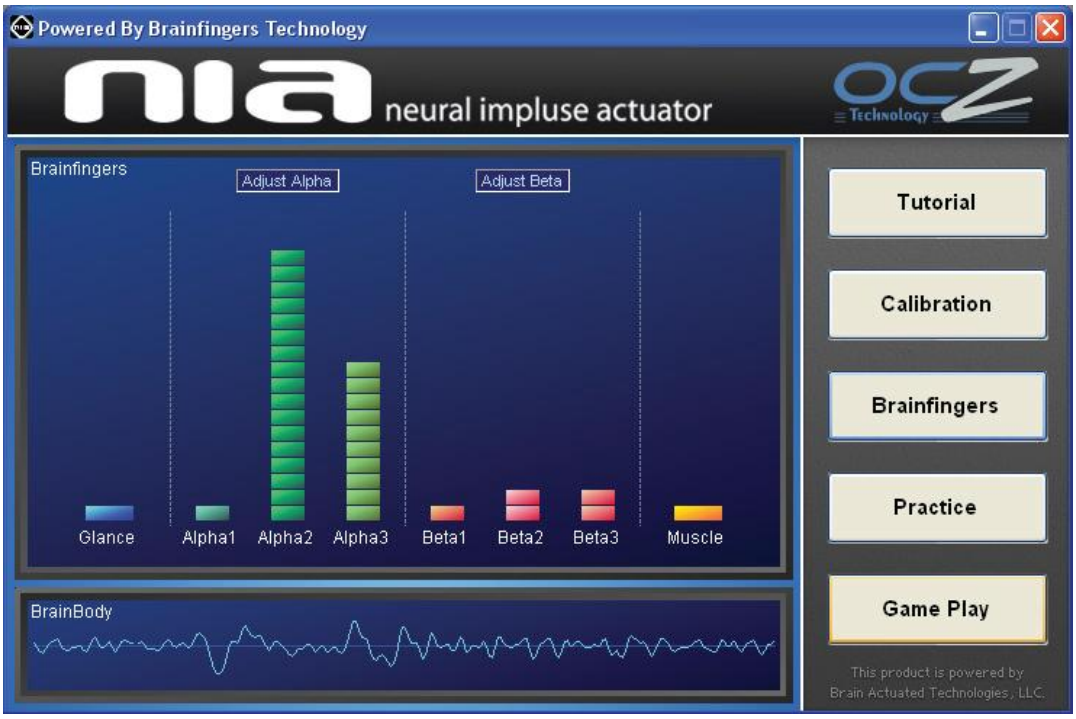

Figura 4-5 Console del software NIA che visualizza in tempo reale le ampiezze dei segnali controllati.

Il programma è stato scritto in C#, che risulta al momento l'unico linguaggio compatibile con l'SDK che ci è stato fornito in versione beta. Per la versione finale del programma, si è però deciso di non utilizzarlo, in primo luogo perché non ancora disponibile ufficialmente, e poi perché l'utilizzo dell'SDK è legato a una licenza della quale non si conosce ancora l'eventuale costo. L'uso di questo ambiente però rende compatibile il programma per future interazioni con l'SDK.

Il programma, oltre alle interfacce appena viste, dispone di una finestra che permette la configurazione dei parametri di comunicazione con la carrozzina e la configurazione dei parametri di utilizzo, come i tempi di visualizzazione delle

immagini, i gradi di rotazione della carrozzina o le destinazioni possibili con le immagini associate. E' inoltre possibile modificare la configurazione attraverso un file XML che contiene tutte queste informazioni. Questo ci permette di avere un set di file da poter copiare sulle varie installazioni senza dover sempre rieditare i parametri tramite l'apposito menù presente nel programma.

Lo schema del file di configurazione è il seguente:

```
<!DOCTYPE Configurazione[
  <!ELEMENT Configurazione (HostName, PortNumber, Timer, TimerAck,
AngoloRotazione, DistanzaMassima, VisualizzaDestinazione,
ListaImmagini)>
  <!ELEMENT HostName (#PCDATA)>
  <!ELEMENT PortNumber (#PCDATA)>
  <!ELEMENT Timer (#PCDATA)>
  <!ELEMENT TimerAck (#PCDATA)>
  <!ELEMENT AngoloRotazione (#PCDATA)>
  <!ELEMENT DistanzaMassima (#PCDATA)>
  <!ELEMENT VisualizzaDestinazione (true|false)>
  <!ELEMENT ListaImmagini (Img+)>
  <!ELEMENT Img EMPTY>
  <!ATTLIST Img Nome CDATA #REQUIRED>
  <!ATTLIST Img Path CDATA #REQUIRED>
]>
```

Descrizione dei campi:

HostName	Indirizzo IP o nome del PC che gestisce la carrozzina.
PortNumber	Porta alla quale collegarsi.
Timer	Tempo in secondi per i quali l'immagine rimane visualizzata, per poi passare alla successiva.
TimerAck	Tempo in secondi per confermare la destinazione.
AngoloRotazione	Angolo di rotazione in gradi usato per le rotazioni.
DistanzaMassima	Distanza massima in metri utilizzata per avanzare o indietreggiare.
VisualizzaDestinazione	Flag che indica se visualizzare oltre all'immagine anche il nome della destinazione in sovraimpressione.
Img Nome	Nome delle destinazione, lo stessa stringa deve essere confiurata sulla carrozzina.
Img Path	Percorso e nome dell'eventuale immagine da visualizzare.

5 Risultati e sviluppi futuri

I risultati ottenuti sono incoraggianti, considerando appunto che si tratta di un dispositivo a basso costo e che questa tecnologia è ancora molto giovane. Il problema maggiore, al momento, riguarda appunto l'alta sensibilità del dispositivo ai disturbi elettromagnetici presenti nell'ambiente. Grazie però alla rilevazione dei soli segnali muscolari che risultano essere molto più precisi e molto meno influenzati dai disturbi rispetto ai segnali cerebrali e oculari, il controllo risulta essere stabile e affidabile.

Lo scopo di questa tesi non è quello di avere un prodotto vendibile, ma quello di dimostrare la possibilità di utilizzare questi dispositivi per applicazioni, non solo ludiche, ma anche socialmente utili. E come detto precedentemente, i risultati sembrano essere incoraggianti. Certamente per poter avere un prodotto commerciale sarà necessario aspettare che la tecnologia diventi più matura, e il passo sembra molto breve, infatti come ci è anche stato riportato dell'inventore del dispositivo, la nuova versione è già in programma e porterà dei miglioramenti tangibili rispetto alla precedente.

Inoltre sarà necessario pensare a dei sistemi di attivazione differenti, non è infatti pensabile utilizzare una modalità come quella sperimentata nella tesi, questo perché qualsiasi movimento muscolare, magari eseguito per poter interagire con altre persone, potrebbe portare all'abilitazione dei comandi sulla carrozzina.

Sarebbe anche interessante riuscire ad eseguire un controllo utilizzando solo le onde cerebrali. Ovviamente utilizzando un dispositivo, sempre retail, ma più affidabile di quello attuale, quindi meno sensibile ai disturbi esterni, e magari

cercando dei sistemi per poter ridurre i tempi di apprendimento, per esempio costruendo dei programmi di allenamento specifici per quella che dovrà poi essere l'applicazione finale.

Un ulteriore sviluppo potrebbe essere quello di riprendere il software di acquisizione diretta dei dati del NIA e rilevare direttamente da questi i movimenti muscolari che, dalle prove che erano state eseguite, sembrerebbero essere la sola componente continua presente nel segnale.

In questo modo sarebbe così possibile eseguire il software direttamente sul computer Linux sul quale funziona già il software di controllo della carrozzina, con un ulteriore abbattimento del prezzo complessivo dell'intero sistema oltre che alla semplificazione hardware del non avere una macchina dedicata alla sola acquisizione dei dati.

Appendice A

MANUALE UTENTE RELATIVO AL PROGRAMMA SVILUPPATO

1. Installazione

   I prerequisiti per poter utilizzare il programma sono l'installazione del software del NIA e l'installazione del framework .net 2.0.

   Il programma è composto solo da un file eseguibile che può essere copiato in una qualsiasi directory del disco, inoltre è possibile copiare nella stessa posizione il file di configurazione config.xml. Se il file non fosse presente è comunque possibile crearlo tramite la finestra di configurazione del programma.

2. Configurazione NIA

   Per poter far interagire il software del NIA con il programma di controllo dei movimenti della carrozzina occorre creare un profilo che mappi i segnali muscolari e oculari in una sequenza di tasti riconosciuti da nostro software. Per fare questo è sufficiente copiare il file Lurch.nia nella directory c:\nia Profiles. Per la modifica e la gestione dei profili si veda la documentazione del NIA.

   Le immagini che seguono riassumono la configurazione del profilo (Figure da A-1 a A-4):

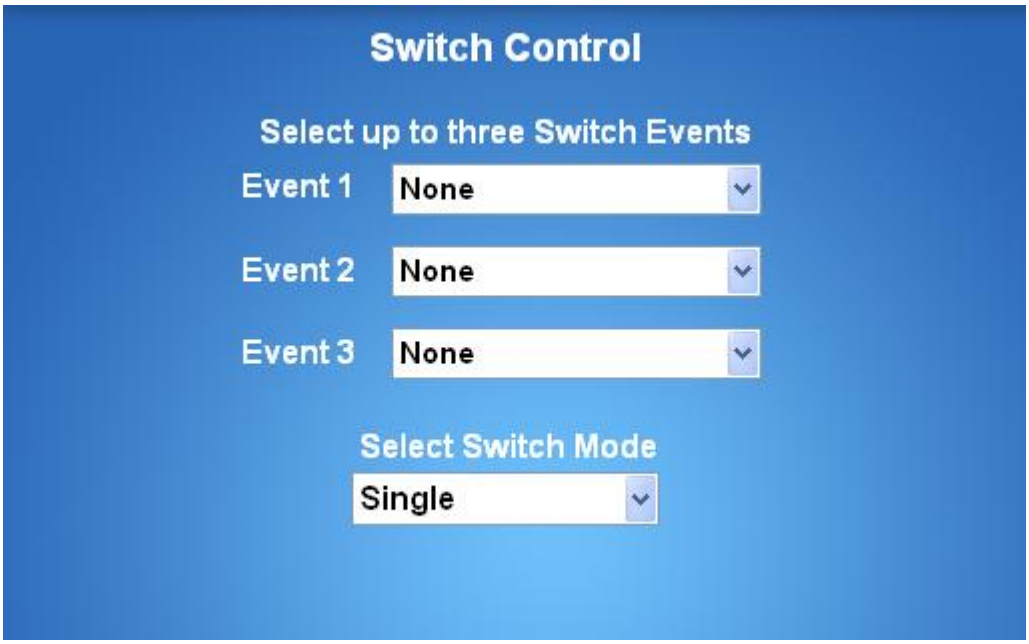

Figura A-1 Selezione delle modalità di comando.

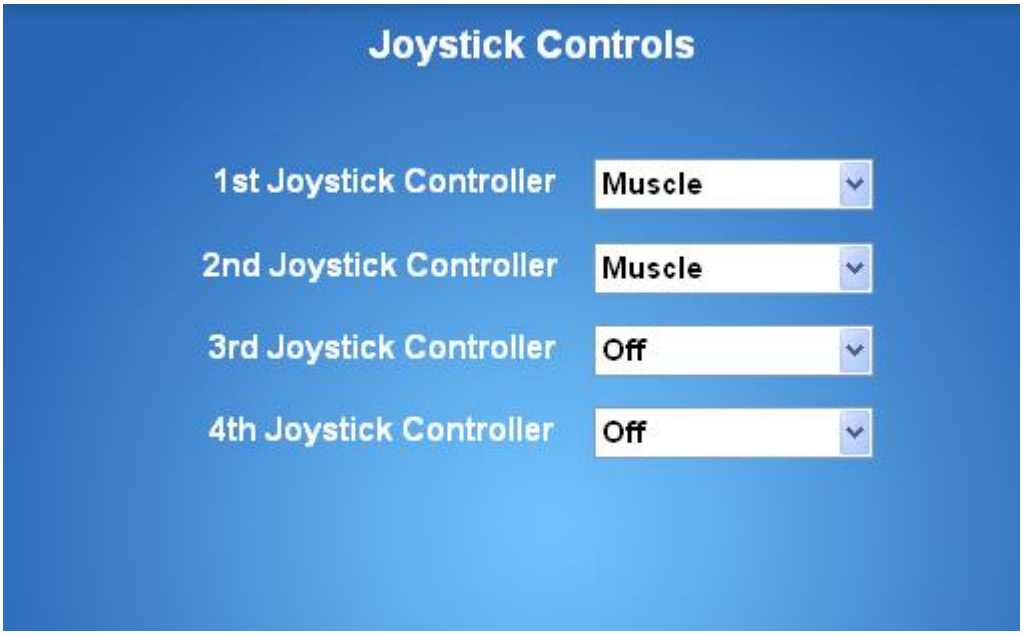

Figura A-2 Selezione dei segnali utilizzati (Muscolare e Oculare).

Figura A-3 Impostazione dei livelli di soglia dei segnali e del tasto che è associato all'evento.

Figura A-4 Impostazione dei tasti associati ai movimenti oculari.

3. Calibrazione NIA

   Una volta installato il software del dispositivo occorrerà eseguire le procedure di calibrazione descritte anch'esse nel manuale d'uso del NIA.

4. Configurazione Software.

   Lanciando l'applicazione viene visualizzata la finestra, Figura A-5, con le modalità disponibili.

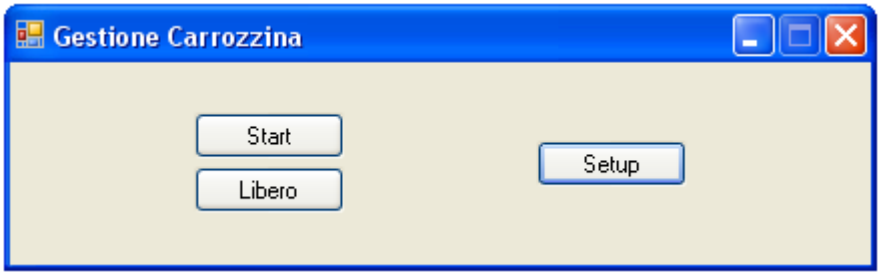

Figura A-5 Finestra di selezione della modalità desiderata.

Per prima cosa occorre configurare il programma cliccando sul tasto Setup. Si aprirà quindi la finestra di configurazione dei parametri del software, Figura A-6.

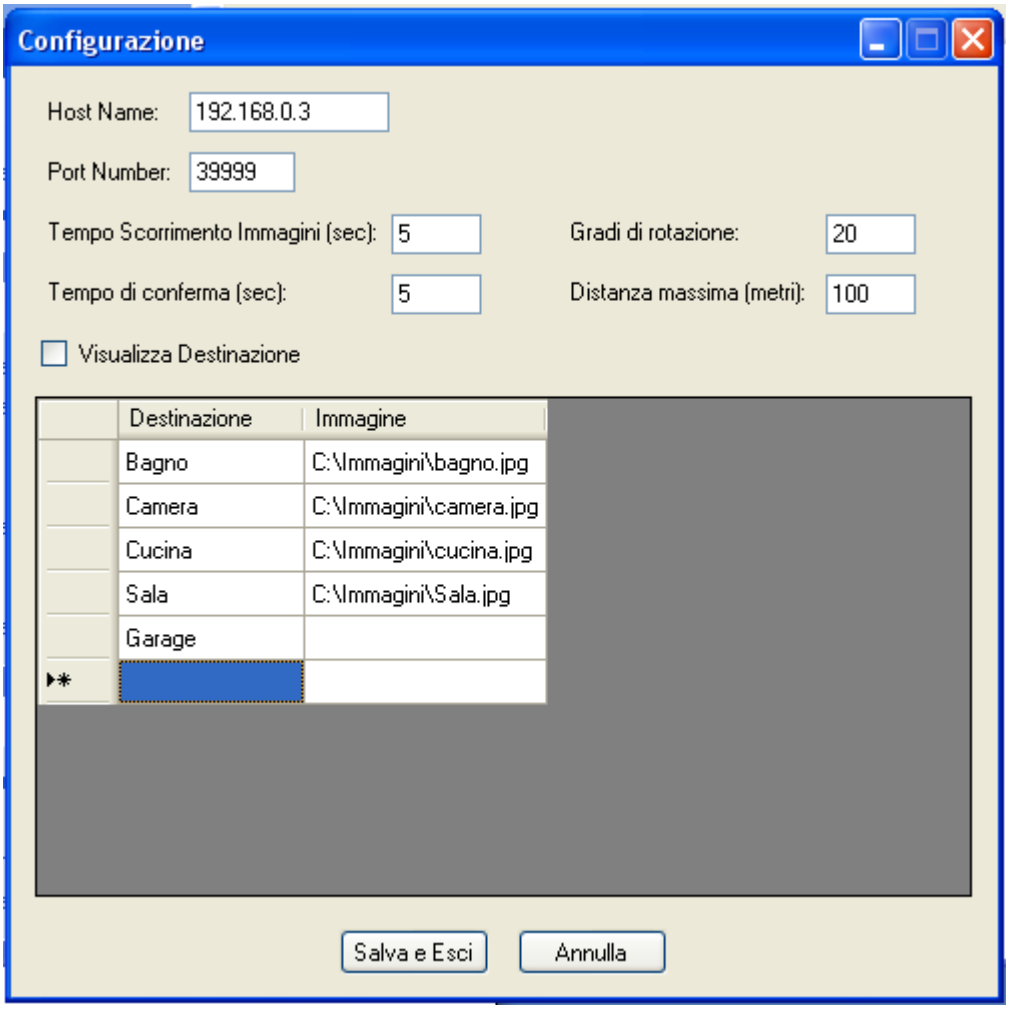

Figura A-6 Finestra di configurazione del software.

In questa finestra è possibile configurare l'indirizzo IP o il nome dell'computer (Host Name) e la porta (Port Number) ai quali il software deve collegarsi per poter interagire con il programma di gestione della carrozzina.

Il campo "Tempo Scorrimento Immagini" è il tempo in secondi che ogni immagine rimane visualizzata per poter essere selezionata dall'utente, mentre il campo "Tempo di conferma", sempre in secondi, rappresenta il tempo che

l'utente ha per confermare la destinazione precedentemente selezionata. E' inoltre possibile selezionare il flag "Visualizza Destinazione" che farà comparire in sovraimpressione il nome della destinazione all'immagine della stessa.

Vi è poi una tabella dove è possibile inserire il nome della destinazione che deve essere esattamente lo stesso che configurato all'interno del software della carrozzina e il path di un'immagine relativa alla destinazione. La selezione dell'immagine non è obbligatoria, nel caso in cui questo campo non sia configurato nella Finestra verrà visualizzato solo il nome della destinazione senza alcuna immagine.

Se si vuole eliminare una destinazione basta selezionarla e premere il tasto "Canc" per cancellarla, se invece si vuole aggiungere una destinazione basta inserirla nell'ultima riga bianca presente in tabella. Infine per selezionare il percorso dove si trova l'immagine basta fare un doppio click sulla casella corrispondente, cosi facendo si aprirà una nuova dialog, Figura A-8, che permetterà la selezione del file.

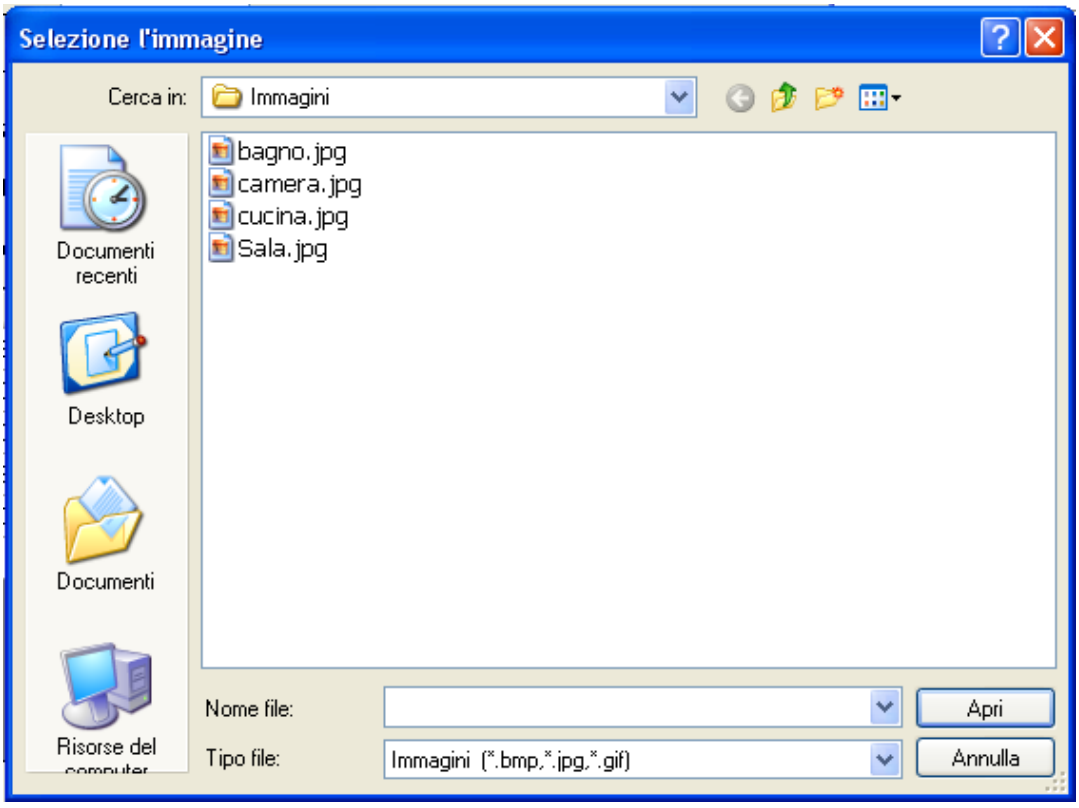

Figura A-7 Finestra di selezione delle immagini relative alle destinazioni.

Infine ci sono due parametri relativi alla funzionalità di spostamento libero, che sono i “gradi di rotazione” che verranno applicati ad ogni comando di rotazione e un campo che indica la distanza “Distanza Metri” che deve essere percorsa al singolo comando di avanzamento. Si ricorda che è possibile interrompere l’avanzamento tramite il medesimo comando, questo campo risulta necessario in quanto non esiste la possibilità di dire al programma di gestione della carrozzina di avanzare fino a un nuovo comando.

5. Esecuzione programma

   Una volta configurato il programma, lanciato il software del NIA con il profilo corretto caricato e in esecuzione (ctrl-f12) e collegato il computer sul quale è in esecuzione il programma con il computer che gestisce la carrozzina, sarà possibile decidere, sempre tramite pannello di controllo se lanciare la modalità di scansione delle destinazioni, pulsante “Start” oppure la modalità di movimento libero, pulsante “Libero”. Nel momento nel quale vengono premuti questi pulsanti, il software proverà a collegarsi al programma di gestione ed eventualmente visualizzerà delle finestre di errore nel caso in cui vi siano dei problemi. In entrambe le modalità si aprirà una nuova Finestra, che sarà possibile chiudere tramite la “X” che si trova in altro a destra sulla finestra.

6. Modalità Automatica

   Quando è in esecuzione questa modalità, attivabile dal tasto “Start”, le immagini delle destinazioni compariranno una alla volta sul monitor. Nel momento in cui viene riconosciuta la destinazione alla quale recarsi, l’utente deve “generare” un movimento muscolare. Nel momento in cui il software riconosce il comando, mostrerà una finestra di richiesta di conferma della destinazione. Nel caso in cui la destinazione è quella corretta, l’utente dovrà generare nuovamente un segnale muscolare entro un tempo configurabile. In caso contrario dovrà semplicemente

attendere che la finestra scompaia e che la sequenza delle destinazioni riprenda a scorrere.

Quando viene confermata la destinazione, il programma rimarrà in attesa che la carrozzina raggiunga la meta prescelta, dopo di che ricomincerà la sequenza delle destinazioni, per poterne selezionare una nuova.

7. Modalità Avanzata

Figura A-8 Finestra di gestione del movimento libero.

Quando è in esecuzione questa modalità, attivabile dal tasto "Libero", l'utente potrà decidere in quale direzione far muovere la carrozzina, Figura A-8.

All'avvio del programma la selezione è già impostata per potersi muovere in avanti, quindi un movimento muscolare di intensità elevata (barra della forza applicata tutta verde) azionerà lo spostamento in avanti della stessa, la quale si arresterà solo a fronte di un analogo segnale, o in alternativa se raggiunge la distanza configurata nel campo "Distanza Metri".

Tramite un movimento muscolare di intensità più bassa (la barra a circa 2/3) sarà

possibile cambiare il comando che verrà eseguito, questo solo se non è in esecuzione un altro comando (ad esempio se la carrozzina sta avanzando).

L'attivazione del comando di retromarcia funziona esattamente come il comando di avanzamento, la carrozzina si sposterà all'indietro finché non si eseguirà un nuovo movimento muscolare della stessa intensità o fino al raggiungimento della distanza massima congiurata.

I comandi di rotazione eseguono una rotazione di un numero di gradi indicati in configurazione, tipicamente 90°. Anche in questo caso se l'angolo di rotazione deve essere minore, tramite l'applicazione di un segnale muscolare di intensità elevata sarà possibile sospendere la rotazione.

Appendice B

SORGENTI DEL PROGRAMMA

Il programma è stato scritto in C# utilizzano l'ambiente Microsoft Visual Studio 2008. Non esiste un solo file contenete tutto il codice, ma vi sono più files che rappresentano le diverse classi del programma e le definizioni delle interfacce grafiche dello stesso. Nel seguito verrà riportato solo il codice generato a mano, con l'eventuale Form associata alla classe, e non quello creato dai wizzard di Visual Studio.

- File Socket.cs. In questo file sono definite le procedure che permettono la comunicazione, utilizzando appunto i socket, tra il nostro software e il software di controllo della carrozzina. Vi è infatti la funzione di connessione, la funzione di invio dei messaggi e quella di ricezione che utilizza un thread e si limita nel indicare tramite un evento e una variabile booleana se è arrivata una risposta e se questa è positiva "OK\n".

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Net.Sockets;
using System.Threading;

namespace GestioneCarrozzina
{
    public class Socket
    {
        [ThreadStatic]
        TcpClient myClient = new TcpClient();
        NetworkStream netStream;
```

```
private String Address;
private Int32 PortNumber;
Thread reciveThread;
public AutoResetEvent autoEvent = new AutoResetEvent(false);
public bool bOk;

private bool bConnect = false;


public Socket(String url, Int32 port)
{
    // Parametri di connessione
    Address = url;
    PortNumber = port;
}

public bool Close()
{
    // Chiusura della conessione
    try
    {
        if (bConnect == true)
        {
            myClient.Close();
            reciveThread.Abort();
            bConnect = false;
        }
        return true;
    }
    catch
    {
        bConnect = false;
        return false;
    }
}

public bool Connect()
{
    // Procedura di conessione
    try
    {
        myClient.Connect(Address, PortNumber);
        netStream = myClient.GetStream();
        reciveThread = new Thread(Recive);
        reciveThread.Start();
        bConnect = true;
        return true;
```

```
            }
            catch
            {
                return false;
            }
        }

        public bool Send(String Message)
        {
            // Invio di un messaggio
            if (bConnect && netStream.CanWrite)
            {
                try
                {
                    Byte[] sendBytes =
Encoding.UTF8.GetBytes(Message);
                    netStream.Write(sendBytes, 0, sendBytes.Length);
                    netStream.Flush();
                    return true;
                }
                catch
                {
                    return false;
                }
            }
            return false;
        }

        public void Recive()
        {
            // Thread per la ricezione delle risposte
            while (true)
            {
                byte[] tmp = new byte[4096];
                int pos = 0;
                try
                {
                    pos += netStream.Read(tmp, pos, 4096-pos);
                    if (pos > 4095)
                        pos = 0;

                    if (pos != 0 && tmp[pos - 1] == 10)
                    {
                        if (pos == 3 && tmp[0] == 'O' && tmp[1] == 'K')
                        {
                            bOk = true;
                        }
```

```
                        else
                        {
                            bOk = false;
                        }
                        pos = 0;
                        autoEvent.Set();
                    }
                }
                catch
                {
                    // Utile per non propagare l'eccezione
                }
            }
        }

    }
}
```

- File Immagini.cs. In questa classe viene gestita tutta la parte di visualizzazione sequenziale delle immagini e il riconoscimento del segnale di selezione, nonché la gestione del caricamento e del ridimensionamento dell'immagine.

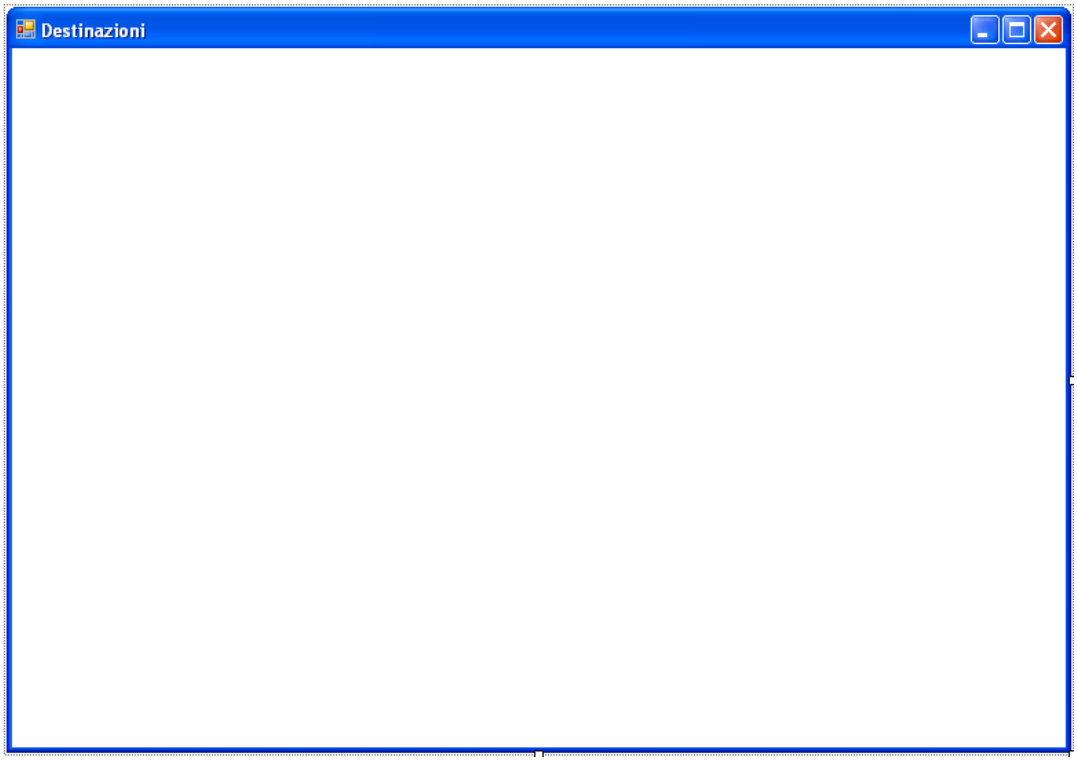

Figura B-1 Dialog nella quale vengono visualizzate le immagini delle destinazioni.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
```

```
using System.Drawing.Drawing2D;
using System.Xml.XPath;

namespace GestioneCarrozzina
{
    public partial class Immagini : Form
    {
        public Socket sock; // Classe per la gestione della cennessione
        private String[] Destinazioni; // Lista delle destinazioni
possibili
        private String[] Path; // Lista delle immagini associate alle
destinazioni
        private int Pos = 0; // Numero destinazione attualmente
visualizzata
        private Principale FinestraPrincipale = null; // Puntatore
alla finestra principale
        bool isImage; // Variabile che indica se disponibile un
immagine per la destinazione attualmente visualizzata
        Image myImage; // Immagine attuale
        String sDestinazione=""; // Destinazione attuale
        int TimerAck = 5; // Tempo di visualizzazione delle immagini
        bool flagTesto = false; // Indica se visualizzare comunque il
nome della destinazione

        public Immagini(Principale pointer)
        {
            InitializeComponent();
            FinestraPrincipale = pointer;
            isImage = false;
        }

        private void LoadImage(int Pos)
        {
            // Router per il caricamento delle immagini
            if (isImage == true)
                myImage.Dispose();

            sDestinazione = Destinazioni[Pos];
            try
            {
                myImage = Image.FromFile(Path[Pos]);
                isImage = true;
            }
            catch
            {
                isImage = false;
            }
```

```
            this.Text = sDestinazione.ToUpper();

            this.Invalidate();
        }

        private void Immagini_Paint(object sender, PaintEventArgs e)
        {
            // Routin che si occupa della visualizzazioni delle
immagini sullo schermo
            if (isImage == true)
            {
                Graphics dc = e.Graphics;
                float r = dc.VisibleClipBounds.Width /
dc.VisibleClipBounds.Height;
                float ri = (float)myImage.Width /
(float)myImage.Height;

                RectangleF destRect;

                if (r > ri)
                {
                    float Width = ri * dc.VisibleClipBounds.Height;
                    destRect = new RectangleF(
(dc.VisibleClipBounds.Width - Width) / 2, 0f, Width,
dc.VisibleClipBounds.Height);
                }
                else
                {
                    float Height = dc.VisibleClipBounds.Width / ri;
                    destRect = new RectangleF(0f,
(dc.VisibleClipBounds.Height - Height) / 2,
dc.VisibleClipBounds.Width, Height);
                }

                dc.InterpolationMode = InterpolationMode.Low;
                dc.SmoothingMode = SmoothingMode.HighSpeed;
                dc.PixelOffsetMode = PixelOffsetMode.HighSpeed;
                dc.CompositingQuality =
CompositingQuality.HighSpeed;
                dc.DrawImage(myImage, destRect);
            }
            if(isImage == false || flagTesto == true)
            {
                using (Font font1 = new Font("Arial", 40,
FontStyle.Bold, GraphicsUnit.Point))
                {
```

```
                    RectangleF rect1 = new RectangleF(0f, 0f,
e.Graphics.VisibleClipBounds.Width,
e.Graphics.VisibleClipBounds.Height);

                    // Imposta il formato della stringa
                    StringFormat stringFormat = new StringFormat();
                    stringFormat.Alignment = StringAlignment.Center;
                    stringFormat.LineAlignment =
StringAlignment.Center;

                    // disegna il testo all'interno dell'area della
form
                    e.Graphics.DrawString(sDestinazione.ToUpper(),
font1, Brushes.Blue, rect1, stringFormat);
                }
            }
        }

        private void Immagini_Resize(object sender, EventArgs e)
        {
            this.Invalidate();
        }

        private void Immagini_KeyPress(object sender,
KeyPressEventArgs e)
        {
            // Gestisce l'evento proveniente dal NIA
            char car = e.KeyChar;
            if (car == 'a')
            {
                timerImmagini.Stop();
                Conferma conf = new Conferma(this,sDestinazione,
TimerAck);
                conf.ShowDialog();
                timerImmagini.Start();
            }
        }

        private void Immagini_Load(object sender, EventArgs e)
        {
            // Carica le impostazioni dal file xml
            int Timer = 5;
            String Name="";
            Int32 Port=0;

            XPathExpression expr;
```

```
            expr =
FinestraPrincipale.xmlNav.Compile("//Configurazione/Timer");
            XPathNavigator node =
FinestraPrincipale.xmlNav.SelectSingleNode(expr);
            if (node != null)
            {
                Timer = int.Parse(node.Value);
            }

            expr =
FinestraPrincipale.xmlNav.Compile("//Configurazione/TimerAck");
            node = FinestraPrincipale.xmlNav.SelectSingleNode(expr);
            if (node != null)
            {
                TimerAck = int.Parse(node.Value);
            }

            expr =
FinestraPrincipale.xmlNav.Compile("//Configurazione/HostName");
            node = FinestraPrincipale.xmlNav.SelectSingleNode(expr);
            if (node != null)
            {
                Name = node.Value;
            }

            expr =
FinestraPrincipale.xmlNav.Compile("//Configurazione/PortNumber");
            node = FinestraPrincipale.xmlNav.SelectSingleNode(expr);
            if (node != null)
            {
                Port = Int32.Parse(node.Value);
            }

            expr =
FinestraPrincipale.xmlNav.Compile("//Configurazione/ListaImmagini/I
mg");
            XPathNodeIterator iterator =
FinestraPrincipale.xmlNav.Select(expr);
            Destinazioni = new String[iterator.Count];
            Path = new String[iterator.Count];
            int x = 0;
            while (iterator.MoveNext())
            {
                node = iterator.Current;
                Destinazioni[x] = node.GetAttribute("Nome","");
                Path[x++] = node.GetAttribute("Path", "");
            }
```

```
            sock = new Socket(Name, Port);
            if (sock.Connect() != true)
            {
                MessageBox.Show("Errore di connessione");
            }

            timerImmagini.Interval = Timer * 1000;
            Pos = 0;
            LoadImage(Pos++);
            timerImmagini.Start();
        }

        private void timerImmagini_Tick(object sender, EventArgs e)
        {
            // Gestione delle temporizzazione delle immagini
            LoadImage(Pos++);
            if (Pos == Destinazioni.GetLength(0))
                Pos = 0;
        }

        private void Immagini_FormClosing(object sender,
FormClosingEventArgs e)
        {
            // Chiusura del programma
            sock.Close();
            timerImmagini.Stop();
        }
    }
}
```

- File Conferma.cs. Questa classe si occupa di gestire la conferma della selezione effettuata in precedenza .

Figura B-2 Finestre usata per la conferma della destinazione.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;

namespace GestioneCarrozzina
{
    public partial class Conferma : Form
    {
        private int Timer;
        private String sDestinazione;
        private Immagini FinestraPadre = null;
```

```
        public Conferma(Immagini pointer, String Destinazione, int
timeout)
        {
            FinestraPadre = pointer;
            Timer = timeout;
            sDestinazione = Destinazione;
            InitializeComponent();
        }

        private void Conferma_Load(object sender, EventArgs e)
        {
            labelTIMER.Text = Timer.ToString();
            textDestinazione.Text = sDestinazione.ToUpper();
            timerAttesa.Interval = 1000;
            timerAttesa.Start();
        }

        private void timerAttesa_Tick(object sender, EventArgs e)
        {
            Timer -= 1;
            labelTIMER.Text = Timer.ToString();
            if (Timer == 0)
                this.Close();
        }

        private void Conferma_KeyPress(object sender,
KeyPressEventArgs e)
        {
            char car = e.KeyChar;
            if ((car == 'a') && (timerAttesaACK.Enabled == false))
            {
                timerAttesa.Stop();
                labelTIMER.Text = "Esecuzione comando in corso...";
                FinestraPadre.sock.Send("destination
"+sDestinazione+"\n");
                label1.Text = "";
                timerAttesaACK.Start();
            }
        }

        private void timerAttesaACK_Tick(object sender, EventArgs e)
        {
            if (FinestraPadre.sock.autoEvent.WaitOne(0,false) ==
true)
            {
                timerAttesaACK.Stop();
                this.Close();
```

```
            }

        }

        private void Conferma_FormClosing(object sender,
FormClosingEventArgs e)
        {
            timerAttesaACK.Stop();
            timerAttesa.Stop();
        }
    }
}
```

- File Pricipale.cs. La classe si occupa della gestione della finestra con i comandi disponibili del software.

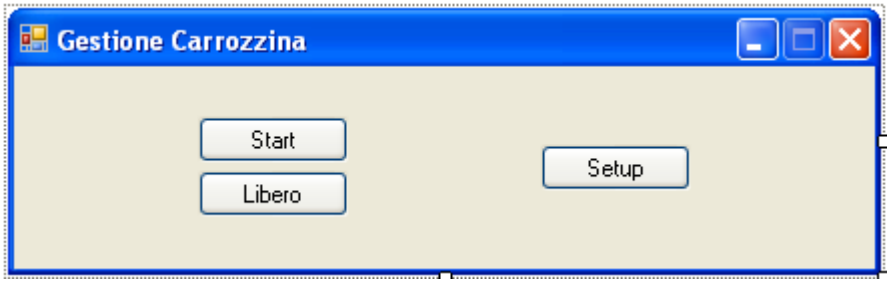

Figura B-3 Finestra principale del programma dalla quale è possibile selezionare la modalità desiderata.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.Xml.XPath;
using System.Xml;

namespace GestioneCarrozzina
{
```

```
    public partial class Principale : Form
    {
        public XmlDocument xmlDoc;
        public XPathNavigator xmlNav;
        public String path;

        public Principale()
        {
            InitializeComponent();
        }

        private void Main_Load(object sender, EventArgs e)
        {
            // Esegue le inizializzanioni del software caricando il
file di configurazione in formato xml
            xmlDoc = new XmlDocument();
            try
            {
                path = Application.StartupPath + "\\Config.xml";
                xmlDoc.Load(path);
            }
            catch
            {
                // Write down the XML declaration
                XmlDeclaration xmlDeclaration =
xmlDoc.CreateXmlDeclaration("1.0", "utf-8", null);

                // Create the root element
                XmlElement rootNode =
xmlDoc.CreateElement("Configurazione");
                xmlDoc.InsertBefore(xmlDeclaration,
xmlDoc.DocumentElement);
                xmlDoc.AppendChild(rootNode);
            }
            xmlNav = xmlDoc.CreateNavigator();
        }

        private void buttonSetup_Click(object sender, EventArgs e)
        {
            // Apre la dialog di configurazione del sistema
            Setup mySetup = new Setup(this);
            mySetup.ShowDialog();
        }

        private void buttonImmagini_Click(object sender, EventArgs e)
        {
            // Lancia la modalità di scorrimento delle destinazioni
```

```
            Immagini myImg = new Immagini(this);
            myImg.ShowDialog();
        }

        private void buttonLibero_Click(object sender, EventArgs e)
        {
            // Lancia la modalità di spostamento libero
            Avanzato myForm = new Avanzato(this);
            myForm.ShowDialog();
        }
    }
}
```

- File Setup.cs. La classe che si occupa della gestione della finestra di configurazione.

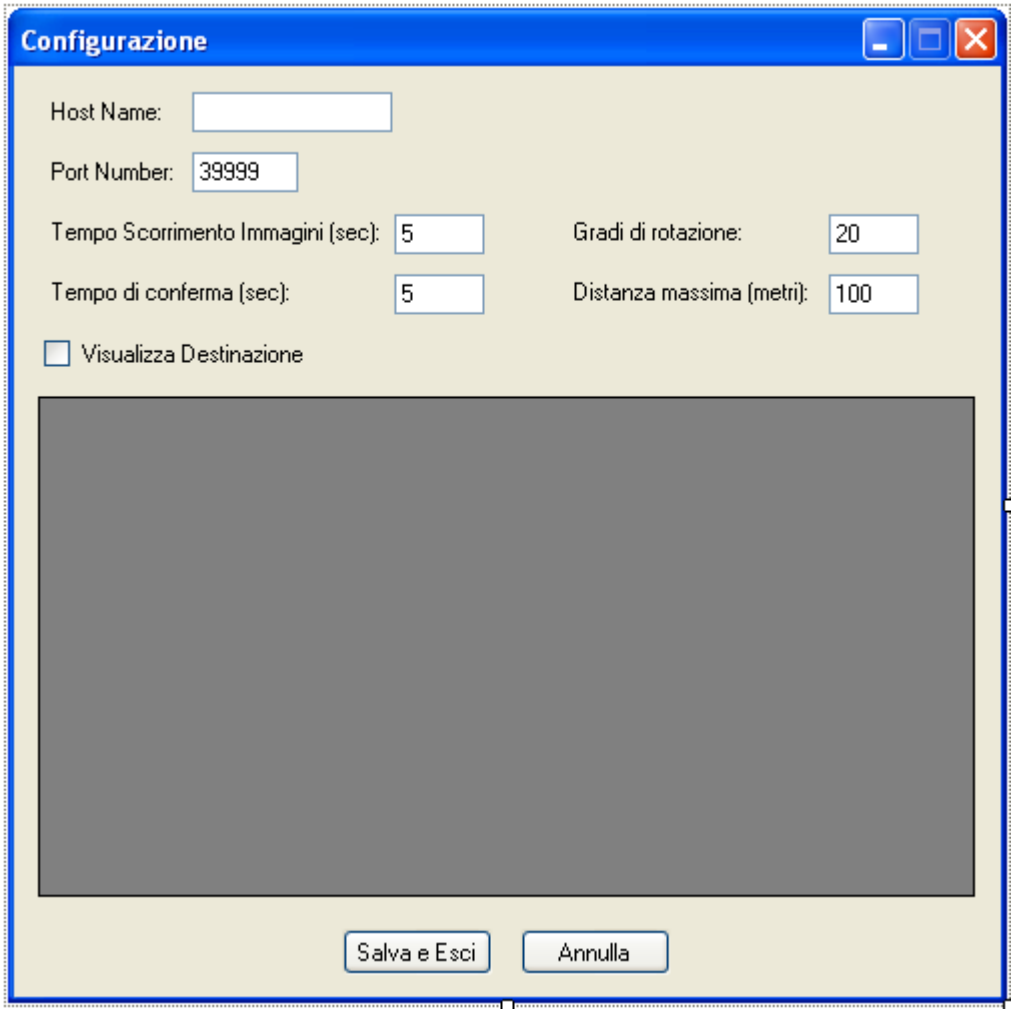

Figura B-4 Finestra di configurazione dei parametri del programma.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
```

```
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.Xml.XPath;
using System.Xml;

namespace GestioneCarrozzina
{
    public partial class Setup : Form
    {
        private Principale FinestraPrincipale = null;

        public Setup(Principale pointer)
        {
            FinestraPrincipale = pointer;
            InitializeComponent();
        }

        private void Setup_Load(object sender, EventArgs e)
        {
            // Inizializzazione dei campi della form dal file xml
            XPathExpression expr;
            XPathNavigator node;

            expr =
FinestraPrincipale.xmlNav.Compile("//Configurazione/Timer");
            node = FinestraPrincipale.xmlNav.SelectSingleNode(expr);
            if (node != null)
            {
                textTimer1.Text = node.Value;
            }

            expr =
FinestraPrincipale.xmlNav.Compile("//Configurazione/TimerAck");
            node = FinestraPrincipale.xmlNav.SelectSingleNode(expr);
            if (node != null)
            {
                textTimer2.Text = node.Value;
            }

            expr =
FinestraPrincipale.xmlNav.Compile("//Configurazione/HostName");
            node = FinestraPrincipale.xmlNav.SelectSingleNode(expr);
            if (node != null)
            {
                textHost.Text = node.Value;
            }
```

```
            expr =
FinestraPrincipale.xmlNav.Compile("//Configurazione/PortNumber");
            node = FinestraPrincipale.xmlNav.SelectSingleNode(expr);
            if (node != null)
            {
                textPort.Text = node.Value;
            }

            expr =
FinestraPrincipale.xmlNav.Compile("//Configurazione/AngoloRotazione
");
            node = FinestraPrincipale.xmlNav.SelectSingleNode(expr);
            if (node != null)
            {
                textGradi.Text = node.Value;
            }

            expr =
FinestraPrincipale.xmlNav.Compile("//Configurazione/DistanzaMassima
");
            node = FinestraPrincipale.xmlNav.SelectSingleNode(expr);
            if (node != null)
            {
                textMetri.Text = node.Value;
            }

            expr =
FinestraPrincipale.xmlNav.Compile("//Configurazione/VisualizzaDesti
nazione");
            node = FinestraPrincipale.xmlNav.SelectSingleNode(expr);
            if (node != null)
            {
                checkBoxDisplay.Checked = bool.Parse(node.Value);
            }

            // Carica i dati delle immagini nella griglia
            XmlDataDocument xmlDatadoc = new XmlDataDocument();
            try
            {

xmlDatadoc.DataSet.ReadXml(FinestraPrincipale.path);
                DataSet ds = new DataSet("Immagini");
                ds = xmlDatadoc.DataSet;
                dataGridView1.DataSource = ds.DefaultViewManager;
                dataGridView1.DataMember = "Img";
            }
```

```
            catch
            {
                DataGridViewTextBoxColumn Dest = new
DataGridViewTextBoxColumn();
                Dest.DataPropertyName = "Destinazione";
                Dest.HeaderText="Destinazione";
                Dest.Name = "Destinazione";
                DataGridViewTextBoxColumn Path = new
DataGridViewTextBoxColumn();
                Path.DataPropertyName = "Path";
                Path.HeaderText = "Immagine";
                Path.Name = "Path";
                dataGridView1.Columns.Insert(0, Dest);
                dataGridView1.Columns.Insert(1, Path);
            }

            DataGridViewTextBoxColumn tmp =
(DataGridViewTextBoxColumn)dataGridView1.Columns[0];
            tmp.HeaderText = "Destinazione";
            tmp.AutoSizeMode =
DataGridViewAutoSizeColumnMode.AllCells;
            tmp =
(DataGridViewTextBoxColumn)dataGridView1.Columns[1];
            tmp.HeaderText = "Immagine";
            tmp.AutoSizeMode =
DataGridViewAutoSizeColumnMode.AllCells;
        }

        private void buttonClose_Click(object sender, EventArgs e)
        {
            // Chiude la Dialog e annulla le modifiche fatte
            this.Close();
        }

        private void buttonSalva_Click(object sender, EventArgs e)
        {
            // salva le impostazioni correnti

            XmlNode root =
FinestraPrincipale.xmlDoc.DocumentElement;
            root.RemoveAll();
            XmlElement childNode =
FinestraPrincipale.xmlDoc.CreateElement("HostName");
            XmlText textNode =
FinestraPrincipale.xmlDoc.CreateTextNode(textHost.Text);
            root.AppendChild(childNode);
            childNode.AppendChild(textNode);
```

```
            childNode =
FinestraPrincipale.xmlDoc.CreateElement("PortNumber");
            textNode =
FinestraPrincipale.xmlDoc.CreateTextNode(textPort.Text);
            root.AppendChild(childNode);
            childNode.AppendChild(textNode);

            childNode =
FinestraPrincipale.xmlDoc.CreateElement("Timer");
            textNode =
FinestraPrincipale.xmlDoc.CreateTextNode(textTimer1.Text);
            root.AppendChild(childNode);
            childNode.AppendChild(textNode);

            childNode =
FinestraPrincipale.xmlDoc.CreateElement("TimerAck");
            textNode =
FinestraPrincipale.xmlDoc.CreateTextNode(textTimer2.Text);
            root.AppendChild(childNode);
            childNode.AppendChild(textNode);

            childNode =
FinestraPrincipale.xmlDoc.CreateElement("AngoloRotazione");
            textNode =
FinestraPrincipale.xmlDoc.CreateTextNode(textGradi.Text);
            root.AppendChild(childNode);
            childNode.AppendChild(textNode);

            childNode =
FinestraPrincipale.xmlDoc.CreateElement("DistanzaMassima");
            textNode =
FinestraPrincipale.xmlDoc.CreateTextNode(textMetri.Text);
            root.AppendChild(childNode);
            childNode.AppendChild(textNode);

            childNode =
FinestraPrincipale.xmlDoc.CreateElement("VisualizzaDestinazione");
            textNode =
FinestraPrincipale.xmlDoc.CreateTextNode(checkBoxDisplay.Checked.To
String());
            root.AppendChild(childNode);
            childNode.AppendChild(textNode);

            childNode =
FinestraPrincipale.xmlDoc.CreateElement("ListaImmagini");
            root.AppendChild(childNode);
```

```
            // Salvataggio immagini
            for (int row = 0; row < dataGridView1.RowCount - 1; row++)
            {
                XmlElement childNodeImg =
FinestraPrincipale.xmlDoc.CreateElement("Img");
                String Destinazione, Path;
                DataGridViewCell cellaDest =
(DataGridViewCell)dataGridView1[0, row];
                DataGridViewCell cellaPath =
(DataGridViewCell)dataGridView1[1, row];
                Destinazione = cellaDest.Value.ToString();
                Path = cellaPath.Value.ToString();
                childNodeImg.SetAttribute("Nome", Destinazione);
                childNodeImg.SetAttribute("Path",Path);
                childNode.AppendChild(childNodeImg);
            }


FinestraPrincipale.xmlDoc.Save(FinestraPrincipale.path);

            this.Close();
        }

        private void dataGridView1_CellDoubleClick(object sender,
DataGridViewCellEventArgs e)
        {
            // Permette la selezione delle immagini
            int col = e.ColumnIndex;
            int riga = e.RowIndex;

            if ((col == 1) && (riga != -1))
            {
                DataGridViewCell cella =
(DataGridViewCell)dataGridView1[col, riga];

                String input = string.Empty;
                OpenFileDialog fDialog = new OpenFileDialog();

                fDialog.Filter = "Immagini
(*.bmp,*.jpg,*.gif)|*.bmp;*.jpg;*.gif|All files (*.*)|*.*";
                fDialog.InitialDirectory = cella.Value.ToString(); ;
                fDialog.Title = "Selezione l'immagine";

                String strFileName = "";
                if (fDialog.ShowDialog() == DialogResult.OK)
                    strFileName = fDialog.FileName;
                if (strFileName == String.Empty)
```

```
                    strFileName = "";
                cella.Value = strFileName;
            }
        }

        #region ControlloEdit
        public bool IsNumeric(string numero) { try {
Int32.Parse(numero); return true; } catch (FormatException) { return
false; } }

        private void textPort_KeyPress(object sender,
KeyPressEventArgs e)
        {
            if (!IsNumeric(e.KeyChar.ToString()) &&
(((int)e.KeyChar) != 8)) e.Handled = true;
        }

        private void textTimer1_KeyPress(object sender,
KeyPressEventArgs e)
        {
            if (!IsNumeric(e.KeyChar.ToString()) &&
(((int)e.KeyChar) != 8)) e.Handled = true;
        }

        private void textTimer2_KeyPress(object sender,
KeyPressEventArgs e)
        {
            if (!IsNumeric(e.KeyChar.ToString()) &&
(((int)e.KeyChar) != 8)) e.Handled = true;
        }

        private void textGradi_KeyPress(object sender,
KeyPressEventArgs e)
        {
            if (!IsNumeric(e.KeyChar.ToString()) &&
(((int)e.KeyChar) != 8)) e.Handled = true;
        }

        private void textMetri_KeyPress(object sender,
KeyPressEventArgs e)
        {
            if (!IsNumeric(e.KeyChar.ToString()) &&
(((int)e.KeyChar) != 8)) e.Handled = true;
        }
        #endregion

    }
```

```
}
```

- File Avanzato.cs. Si occupa della gestione dei comandi di movimento libero della carrozzina.

Figura B-5 Finestra visualizzata durante il funzionamento nella modalità di movimentazione libera.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.Xml.XPath;
using System.Media;

namespace GestioneCarrozzina
{
    public partial class Avanzato : Form
```

```
    {
        // Stati possibili
        enum Stato { Fermo, Avanti, Destra, Sinistra, Indietro };

        // Puntatore alla classe di gestione della connessione
        public Socket sock;

        private Principale FinestraPrincipale = null;
        private int StatoCorrente = (int) Stato.Fermo; // Operazione
attualmente in corso
        private int DirezioneSelezionata = (int)Stato.Avanti;
        private int GradiRotazione = 20;
        private int DistanzaMetri = 100;

        public Avanzato(Principale pointer)
        {
            InitializeComponent();
            FinestraPrincipale = pointer;
        }

        private void Avanzato_Load(object sender, EventArgs e)
        {
            StatoCorrente = (int)Stato.Fermo;
            String Name = "";
            Int32 Port = 0;

            XPathExpression expr;

            expr =
FinestraPrincipale.xmlNav.Compile("//Configurazione/HostName");
            XPathNavigator node =
FinestraPrincipale.xmlNav.SelectSingleNode(expr);
            if (node != null)
            {
                Name = node.Value;
            }

            expr =
FinestraPrincipale.xmlNav.Compile("//Configurazione/PortNumber");
            node = FinestraPrincipale.xmlNav.SelectSingleNode(expr);
            if (node != null)
            {
                Port = Int32.Parse(node.Value);
            }
```

```
            expr =
FinestraPrincipale.xmlNav.Compile("//Configurazione/DistanzaMassima
");
            node = FinestraPrincipale.xmlNav.SelectSingleNode(expr);
            if (node != null)
            {
                DistanzaMetri = Int32.Parse(node.Value);
            }

            expr =
FinestraPrincipale.xmlNav.Compile("//Configurazione/AngoloRotazione
");
            node = FinestraPrincipale.xmlNav.SelectSingleNode(expr);
            if (node != null)
            {
                GradiRotazione = Int32.Parse(node.Value);
            }

            sock = new Socket(Name, Port);
            if (sock.Connect() != true)
            {
                MessageBox.Show("Errore di connessione");
                this.Text = "Sistema non connesso...";
            }

        }

        private void Avanzato_FormClosing(object sender,
FormClosingEventArgs e)
        {
            sock.Close();
            timerSelezione.Stop();
        }

        private void Avanzato_KeyPress(object sender,
KeyPressEventArgs e)
        {
            char car = e.KeyChar;
            switch (car)
            {
                case '1':
                    progressBarForza.Value = 1;
                    break;
                case '2':
                    progressBarForza.Value = 2;
                    break;
                case '3':
```

```
                progressBarForza.Value = 3;
                break;
            case '4':
                progressBarForza.Value = 4;
                break;
            case '5':
                progressBarForza.Value = 4;
                break;
            case '6':
                if (progressBarForza.Value == 4)
                {
                    SystemSounds.Beep.Play();
                    timerSelezione.Start();
                }
                progressBarForza.Value = 5;
                break;
            case '7':
                progressBarForza.Value = 6;
                break;
            case 'a':
                timerSelezione.Stop();
                progressBarForza.Value = 7;
                switch (DirezioneSelezionata)
                {
                    case (int)Stato.Avanti:
                        Avanti();
                        break;
                    case (int)Stato.Indietro:
                        Indietro();
                        break;
                    case (int)Stato.Destra:
                        Destra();
                        break;
                    case (int)Stato.Sinistra:
                        Sinistra();
                        break;
                }

                break;
        }
    }

    private void EvidenziaDirezione()
    {
        String sDirezione = "Avanti";
        switch (DirezioneSelezionata)
        {
```

```
                case (int)Stato.Avanti:
                    sDirezione = "Avanti";
                    buttonAvanti.Select();
                    break;
                case (int)Stato.Indietro:
                    sDirezione = "Indietro";
                    buttonIndietro.Select();
                    break;
                case (int)Stato.Destra:
                    sDirezione = "Destra";
                    buttonDestra.Select();
                    break;
                case (int)Stato.Sinistra:
                    sDirezione = "Sinistra";
                    buttonSinistra.Select();
                    break;
            }
            textBoxDirezione.Text = sDirezione;
        }

        private void Avanti()
        {
            if (StatoCorrente != (int)Stato.Fermo)
            {
                timer1.Stop();
                sock.Send("stop\n");
                if (sock.autoEvent.WaitOne(2000, false) == false)
                {
                    System.Console.WriteLine("Erorre Ack");
                }
                StatoCorrente = (int)Stato.Fermo;
            }
            else
            {
                String Comando = "step_tan " +
DistanzaMetri.ToString() + "\n";
                timer1.Start();
                StatoCorrente = (int)Stato.Avanti;
                sock.Send(Comando);
            }
        }

        private void Indietro()
        {
            if (StatoCorrente != (int)Stato.Fermo)
            {
                timer1.Stop();
```

```
                sock.Send("stop\n");
                if (sock.autoEvent.WaitOne(2000, false) == false)
                {
                    System.Console.WriteLine("Erorre Ack");
                }
                StatoCorrente = (int)Stato.Fermo;
            }
            else
            {
                String Comando = "step_tan -" +
DistanzaMetri.ToString() + "\n";
                timer1.Start();
                StatoCorrente = (int)Stato.Indietro;
                sock.Send(Comando);
            }
        }

        private void Destra()
        {
            if (StatoCorrente != (int)Stato.Fermo)
            {
                timer1.Stop();
                sock.Send("stop\n");
                if (sock.autoEvent.WaitOne(2000, false) == false)
                {
                    System.Console.WriteLine("Erorre Ack");
                }
                StatoCorrente = (int)Stato.Fermo;
            }
            else
            {
                String Comando = "step_rot -" +
GradiRotazione.ToString() + "\n";
                timer1.Start();
                StatoCorrente = (int)Stato.Destra;
                sock.Send(Comando);
            }
        }

        private void Sinistra()
        {
            if (StatoCorrente != (int)Stato.Fermo)
            {
                timer1.Stop();
                sock.Send("stop\n");
                if (sock.autoEvent.WaitOne(2000, false) == false)
                {
```

```
                    System.Console.WriteLine("Erorre Ack");
                }
                StatoCorrente = (int)Stato.Fermo;
            }
            else
            {
                String Comando = "step_rot " +
GradiRotazione.ToString() + "\n";
                timer1.Start();
                StatoCorrente = (int)Stato.Sinistra;
                sock.Send(Comando);
            }
        }

        private void timer1_Tick(object sender, EventArgs e)
        {
            if (sock.autoEvent.WaitOne(0, false) == true)
            {
                StatoCorrente = (int)Stato.Fermo;
            }
        }

        private void timerSelezione_Tick(object sender, EventArgs e)
        {
            timerSelezione.Stop();
            if (StatoCorrente == (int)Stato.Fermo)
            {
                DirezioneSelezionata++;
                if (DirezioneSelezionata > (int)Stato.Indietro)
                    DirezioneSelezionata = (int)Stato.Avanti;
                EvidenziaDirezione();
            }
        }
    }
}
```

- Viene riportato un esempio di file di configurazione Config.xml.

```
<?xml version="1.0" encoding="utf-8"?>
<Configurazione>
  <HostName>192.168.0.3</HostName>
  <PortNumber>39999</PortNumber>
  <Timer>5</Timer>
  <TimerAck>5</TimerAck>
  <AngoloRotazione>90</AngoloRotazione>
  <DistanzaMassima>100</DistanzaMassima>
  <VisualizzaDestinazione>False</VisualizzaDestinazione>
  <ListaImmagini>
    <Img Nome="bagno" Path="c:\Immagini\camera.jpg" />
    <Img Nome="camera" Path="c:\Immagini\camera.jpg" />
    <Img Nome="cucina" Path="c:\Immagini\cucina.jpg" />
    <Img Nome="sala" Path="c:\Immagini\Sala.jpg" />
  </ListaImmagini>
</Configurazione>
```

- Per completezza viene di seguito riportato anche il file di configurazione del NIA, Lurch.nia.

```
JS2_TR3_E3_HOLD_TIME=0
JS4_TR3_E1_HOLD_TIME=0
GAME_EXE_PATH=
GLANCE_COMPUTER_JOYSTICK_SMOOTH=3.0
JS2_TR3_E3_DWELL_TIME=0
JS4_TR3_EVENT_3=None
JS3_TR4_MODE=Single
PONG_BALL_SIZE=50
JS3_TR2_EVENT_2=None
JS2_TR1_EVENT_1=6
JS1_TR3_E3_REPEAT_TIME=0
JS3_TR4_E3_REPEAT_TIME=0
JS2_TR2_E1_DWELL_TIME=0
JS2_TR1_VALUE=51
EMG_SCALE=10
JS2_TR1_E1_HOLD_TIME=0
JS3_TR1_VALUE=999
JS2_TR3_E3_REPEAT_TIME=0
JS4_TR4_E2_DWELL_TIME=0
JS4_TR3_E1_REPEAT_TIME=0
```

```
JS4_TR3_E3_REPEAT_TIME=0
JS1_TR2_E3_DWELL_TIME=0
JS1_TR4_E3_HOLD_TIME=0
JS4_TR1_E2_HOLD_TIME=0
JS3_TR4_EVENT_2=None
BUTTON_COLOR_R=215
JS3_TR4_E1_HOLD_TIME=0
JS1_TR3_EVENT_1=3
JS2_TR2_E1_REPEAT_TIME=0
JS2_TR1_EVENT_3=None
BALANCE_AAC_ON_WHEN_CTRLF12_ON=1
JS3_TR3_EVENT_2=None
JS4_TR2_E2_DWELL_TIME=0
JS4_TR1_E2_REPEAT_TIME=0
JS1_TR4_VALUE=51
JS2_TR1_E3_HOLD_TIME=0
TITLE_STARTUP_PROFILE=StartUp
JS4_TR2_E3_DWELL_TIME=0
JS2_TR3_E1_DWELL_TIME=0
JS2_TR1_E3_DWELL_TIME=0
BALANCE_STEP_MINUS=7
BALANCE_MODE=1
JS4_TR2_E1_HOLD_TIME=0
JS1_TR3_E2_HOLD_TIME=0
JS1_TR2_EVENT_1=2
JS4_TR1_E3_DWELL_TIME=0
JS1_TR1_E2_REPEAT_TIME=0
JS3_TR4_E3_HOLD_TIME=0
JS4_TR1_E1_HOLD_TIME=0
JS4_TR3_E3_DWELL_TIME=0
JS2_TR1_E3_REPEAT_TIME=0
JS3_TR1_EVENT_1=None
SW_E3_REPEAT_TIME=0
JS1_TR4_EVENT_1=4
JS3_TR1_E2_REPEAT_TIME=0
JS2_TR3_VALUE=78
JS1_TR1_E3_DWELL_TIME=0
JS2_TR4_E2_HOLD_TIME=0
CH_FOR_JOYSTICK_2=8
JS4_TR2_EVENT_2=None
JS3_TR1_E1_REPEAT_TIME=0
JS2_TR4_E1_HOLD_TIME=0
JS3_TR4_E3_DWELL_TIME=0
BALANCE_SHIFT=2
JS1_TR1_EVENT_1=1
JS4_TR4_MODE=Single
JS3_TR3_E3_DWELL_TIME=0
```

```
JS3_TR2_E3_DWELL_TIME=0
TIME_OF_SIGNAL_ADJUST=15/02/2010 10.42.31
JS2_TR1_E1_DWELL_TIME=0
JS1_TR2_E2_HOLD_TIME=0
JS2_TR3_E2_DWELL_TIME=0
SW_E1_REPEAT_TIME=0
JS2_TR3_E1_REPEAT_TIME=0
JS3_TR2_EVENT_3=None
JS2_TR3_EVENT_2=None
JS3_TR2_E2_DWELL_TIME=0
HEADBAND_DIRECTION=1
USING_NIA=0
JS1_TR2_E1_REPEAT_TIME=0
PONG_PROFILE_TYPE=1
JS1_TR2_EVENT_3=None
JS1_TR1_EVENT_3=None
JS4_TR4_E2_HOLD_TIME=0
JS1_TR3_MODE=Single
SW_EVENT_3=None
JS1_TR2_E2_DWELL_TIME=0
JS3_TR2_E1_DWELL_TIME=0
PONG_BALL_SPEED=35
GLANCE_MAG_THRESHOLD=60
JS3_TR4_E1_DWELL_TIME=0
EOG_SCALE=10
JS1_TR1_E3_REPEAT_TIME=0
JS1_TR1_E2_DWELL_TIME=0
JS3_TR2_MODE=Single
HIGH_MUSCLE_CUTOFF_RATIO=10.0
BALANCE_CUTOFF=11
JS_VT_AUDIO=0
JS3_TR1_E2_HOLD_TIME=0
JS3_TR1_E3_REPEAT_TIME=0
JS4_TR1_EVENT_2=None
BALANCE_OFF_WHEN_CTRLF12_ON=1
LIMIT_MOTOR_UNIT_PEAKS=0
JS1_TR4_E2_HOLD_TIME=0
JS2_TR1_E2_DWELL_TIME=0
JS2_TR1_EVENT_2=None
JS1_TR1_E2_HOLD_TIME=0
JS2_TR4_EVENT_1=A
JS1_TR1_E1_DWELL_TIME=0
GLANCE_PH_THRESHOLD=600
JS2_TR2_E3_HOLD_TIME=0
JS3_TR4_E1_REPEAT_TIME=0
JS4_TR2_EVENT_3=None
JS4_TR4_E1_REPEAT_TIME=0
```

```
JS3_TR3_E2_REPEAT_TIME=0
JS1_TR4_E1_DWELL_TIME=0
JS1_TR3_EVENT_3=None
JS2_TR2_EVENT_1=6
JS2_TR4_MODE=Single
JS3_TR1_E3_DWELL_TIME=0
JS2_TR2_EVENT_3=None
JS2_TR4_E2_REPEAT_TIME=0
JS1_TR4_E2_REPEAT_TIME=0
JS1_TR1_MODE=Single
JS1_TR4_MODE=Single
JS3_TR4_E2_DWELL_TIME=0
JS4_TR2_VALUE=999
JS3_TR2_E2_HOLD_TIME=0
JS4_TR4_EVENT_2=None
JS3_TR3_E2_DWELL_TIME=0
BUTTON_COLOR_B=255
JS1_TR2_E3_REPEAT_TIME=0
JS1_TR3_E2_REPEAT_TIME=0
JS1_TR3_E3_DWELL_TIME=0
JS3_TR3_VALUE=999
EEG_SCALE=10
SW_MODE=Single
JS4_TR2_E3_HOLD_TIME=0
JS2_TR2_E3_DWELL_TIME=0
JS4_TR1_E1_REPEAT_TIME=0
JS3_TR1_EVENT_3=None
SW_E3_HOLD_TIME=0
SW_EVENT_1=None
JS4_TR3_E1_DWELL_TIME=0
JS4_TR3_MODE=Single
JS3_TR2_E1_REPEAT_TIME=0
JS2_TR3_MODE=Single
BUTTON_COLOR_G=225
JS4_TR3_E2_REPEAT_TIME=0
JS3_TR2_E2_REPEAT_TIME=0
JS2_TR3_E2_HOLD_TIME=0
JS3_TR1_E1_DWELL_TIME=0
JS1_TR1_E1_REPEAT_TIME=0
JS1_TR4_E1_HOLD_TIME=0
JS2_TR4_EVENT_3=None
JS4_TR2_EVENT_1=None
JS4_TR1_E3_HOLD_TIME=0
JS2_TR2_MODE=Single
JS1_TR3_E1_HOLD_TIME=0
JS4_TR1_E1_DWELL_TIME=0
JS1_TR2_EVENT_2=None
```

```
JS4_TR2_E1_REPEAT_TIME=0
JS4_TR1_MODE=Single
JS4_TR3_EVENT_1=None
JS4_TR2_E2_REPEAT_TIME=0
SW_SOUND_ON=1
JS1_TR3_E1_DWELL_TIME=0
BALANCE_HISTORY=2
JS4_TR4_E3_REPEAT_TIME=0
JS1_TR4_EVENT_2=None
JS1_TR3_VALUE=34
JS1_TR2_MODE=Single
JS1_TR4_E3_DWELL_TIME=0
JS4_TR3_E2_HOLD_TIME=0
JS2_TR4_E3_REPEAT_TIME=0
JS1_TR4_E3_REPEAT_TIME=0
JS2_TR4_E1_DWELL_TIME=0
JS1_TR3_E3_HOLD_TIME=0
JS1_TR1_EVENT_2=None
JS4_TR1_EVENT_3=None
JS3_TR3_E1_REPEAT_TIME=0
JS2_TR1_E2_HOLD_TIME=0
JS2_TR4_E3_DWELL_TIME=0
JS1_TR2_E1_HOLD_TIME=0
JS3_TR1_E2_DWELL_TIME=0
JS3_TR4_VALUE=999
JS2_TR4_E3_HOLD_TIME=0
JS2_TR3_EVENT_1=7
JS2_TR1_E1_REPEAT_TIME=0
JS3_TR4_EVENT_1=None
JS4_TR1_E2_DWELL_TIME=0
JS3_TR4_E2_HOLD_TIME=0
JS4_TR1_E3_REPEAT_TIME=0
GLANCE_COMPUTER_JOYSTICK_SHIFT=10.0
JS3_TR3_EVENT_1=None
JS3_TR2_E3_REPEAT_TIME=0
JS1_TR3_E2_DWELL_TIME=0
HIGH_MUSCLE_CUTOFF_ON=0
SW_EVENT_2=None
JS2_TR4_VALUE=90
BALANCE_STEP_PLUS=7
JS2_TR2_E2_DWELL_TIME=0
JS1_TR2_E3_HOLD_TIME=0
SKILL_LEVEL_COMPUTER=20
JS_HZ_AUDIO=0
JS4_TR4_E1_HOLD_TIME=0
JS4_TR2_MODE=Single
JS1_TR1_VALUE=1
```

```
GLANCE_COMPUTER_JOYSTICK_AMPLIFY=5.0
JS2_TR1_MODE=Single
JS3_TR4_EVENT_3=None
JS4_TR2_E3_REPEAT_TIME=0
PONG_CHANNEL_UP_DN=8
JS1_TR3_EVENT_2=None
JS3_TR1_MODE=Single
JS3_TR1_E1_HOLD_TIME=0
PONG_PADDLE_SIZE=60
JS3_TR3_EVENT_3=None
JS4_TR4_E3_HOLD_TIME=0
JS2_TR2_EVENT_2=None
CH_FOR_JOYSTICK_3=0
CH_FOR_JOYSTICK_1=8
JS1_TR1_E1_HOLD_TIME=0
CH_FOR_JOYSTICK_4=0
JS1_TR2_E1_DWELL_TIME=0
JS2_TR2_E2_REPEAT_TIME=0
JS4_TR3_EVENT_2=None
PONG_BALLS_PER_GAME=10
JS4_TR1_EVENT_1=None
JS3_TR3_MODE=Single
JS3_TR3_E2_HOLD_TIME=0
JS2_TR4_E1_REPEAT_TIME=0
JS1_TR4_EVENT_3=None
JS1_TR4_E1_REPEAT_TIME=0
JS4_TR4_E2_REPEAT_TIME=0
JS3_TR2_EVENT_1=None
JS4_TR4_E3_DWELL_TIME=0
JS3_TR1_E3_HOLD_TIME=0
JS1_TR4_E2_DWELL_TIME=0
JS3_TR1_EVENT_2=None
JS4_TR2_E1_DWELL_TIME=0
JS1_TR3_E1_REPEAT_TIME=0
JS1_TR1_E3_HOLD_TIME=0
EOG_OFFSET=0
JS3_TR4_E2_REPEAT_TIME=0
JS3_TR2_E1_HOLD_TIME=0
JS4_TR2_E2_HOLD_TIME=0
JS4_TR3_E3_HOLD_TIME=0
JS4_TR4_EVENT_1=None
JS3_TR3_E1_HOLD_TIME=0
JS2_TR3_E2_REPEAT_TIME=0
JS2_TR4_EVENT_2=None
JS2_TR4_E2_DWELL_TIME=0
JS1_TR2_VALUE=17
JS2_TR2_E3_REPEAT_TIME=0
```

```
JS4_TR3_VALUE=999
SW_E1_HOLD_TIME=0
JS1_TR2_E2_REPEAT_TIME=0
JS2_TR2_E2_HOLD_TIME=0
TIME_OF_CALIBRATION=15/02/2010 11.29.26
JS4_TR4_VALUE=999
CONTROL_DESKTOP_ON=0
JS3_TR2_VALUE=999
SW_E2_REPEAT_TIME=0
JS2_TR2_E1_HOLD_TIME=0
JS4_TR4_E1_DWELL_TIME=0
JS3_TR2_E3_HOLD_TIME=0
JS2_TR3_E1_HOLD_TIME=0
SWITCH_AUDIO=0
JS3_TR3_E3_HOLD_TIME=0
SW_E2_HOLD_TIME=0
JS3_TR3_E1_DWELL_TIME=0
JS4_TR3_E2_DWELL_TIME=0
PONG_CHANNEL_LF_RT=1
CH_FOR_SWITCH=0
JS2_TR2_VALUE=65
JS4_TR1_VALUE=999
JS3_TR3_E3_REPEAT_TIME=0
JS2_TR3_EVENT_3=None
JS2_TR1_E2_REPEAT_TIME=0
JS4_TR4_EVENT_3=None
CH0+SWITCH_SHIFT=-175
CH0+DISPLAY_SMOOTH=77
CH0+JOYSTICK_AMPLIFY=8,5
CH0+SWITCH_AMPLIFY=5
CH0+JOYSTICK_SHIFT=6,7
CH0+DISPLAY_SHIFT=-175
CH0+JOYSTICK_SMOOTH=80
CH0+DISPLAY_AMPLIFY=5
CH0+BASELINE_SHIFT=728
CH0+SWITCH_SMOOTH=10
CH9+SWITCH_SHIFT=-175
CH9+DISPLAY_SMOOTH=77
CH9+JOYSTICK_AMPLIFY=5
CH9+SWITCH_AMPLIFY=5
CH9+JOYSTICK_SHIFT=-175
CH9+DISPLAY_SHIFT=-175
CH9+JOYSTICK_SMOOTH=80
CH9+DISPLAY_AMPLIFY=5
CH9+BASELINE_SHIFT=52
CH9+SWITCH_SMOOTH=10
CH1+SWITCH_SHIFT=-175
```

```
CH1+DISPLAY_SMOOTH=77
CH1+CENTER_FREQUENCY_HZ=8
CH1+JOYSTICK_AMPLIFY=5
CH1+SWITCH_AMPLIFY=5
CH1+JOYSTICK_SHIFT=-175
CH1+DISPLAY_SHIFT=-175
CH1+JOYSTICK_SMOOTH=80
CH1+DISPLAY_AMPLIFY=5
CH1+BASELINE_SHIFT=15
CH1+SWITCH_SMOOTH=10
CH2+SWITCH_SHIFT=-175
CH2+DISPLAY_SMOOTH=77
CH2+CENTER_FREQUENCY_HZ=10
CH2+JOYSTICK_AMPLIFY=5
CH2+SWITCH_AMPLIFY=5
CH2+JOYSTICK_SHIFT=-175
CH2+DISPLAY_SHIFT=-175
CH2+JOYSTICK_SMOOTH=80
CH2+DISPLAY_AMPLIFY=5
CH2+BASELINE_SHIFT=15
CH2+SWITCH_SMOOTH=10
CH3+SWITCH_SHIFT=-175
CH3+DISPLAY_SMOOTH=77
CH3+CENTER_FREQUENCY_HZ=12
CH3+JOYSTICK_AMPLIFY=5
CH3+SWITCH_AMPLIFY=5
CH3+JOYSTICK_SHIFT=-175
CH3+DISPLAY_SHIFT=-175
CH3+JOYSTICK_SMOOTH=80
CH3+DISPLAY_AMPLIFY=5
CH3+BASELINE_SHIFT=15
CH3+SWITCH_SMOOTH=10
CH4+SWITCH_SHIFT=-175
CH4+DISPLAY_SMOOTH=77
CH4+CENTER_FREQUENCY_HZ=16
CH4+JOYSTICK_AMPLIFY=5
CH4+SWITCH_AMPLIFY=5
CH4+JOYSTICK_SHIFT=-175
CH4+DISPLAY_SHIFT=-175
CH4+JOYSTICK_SMOOTH=80
CH4+DISPLAY_AMPLIFY=5
CH4+BASELINE_SHIFT=10
CH4+SWITCH_SMOOTH=10
CH5+SWITCH_SHIFT=-175
CH5+DISPLAY_SMOOTH=77
CH5+CENTER_FREQUENCY_HZ=20
CH5+JOYSTICK_AMPLIFY=7,2
```

```
CH5+SWITCH_AMPLIFY=5
CH5+JOYSTICK_SHIFT=-147
CH5+DISPLAY_SHIFT=-175
CH5+JOYSTICK_SMOOTH=80
CH5+DISPLAY_AMPLIFY=5
CH5+BASELINE_SHIFT=10
CH5+SWITCH_SMOOTH=10
CH6+SWITCH_SHIFT=-175
CH6+DISPLAY_SMOOTH=77
CH6+CENTER_FREQUENCY_HZ=24
CH6+JOYSTICK_AMPLIFY=5
CH6+SWITCH_AMPLIFY=5
CH6+JOYSTICK_SHIFT=-175
CH6+DISPLAY_SHIFT=-175
CH6+JOYSTICK_SMOOTH=80
CH6+DISPLAY_AMPLIFY=5
CH6+BASELINE_SHIFT=10
CH6+SWITCH_SMOOTH=10
CH7+SWITCH_SHIFT=-175
CH7+DISPLAY_SMOOTH=77
CH7+JOYSTICK_AMPLIFY=0,6
CH7+SWITCH_AMPLIFY=5
CH7+JOYSTICK_SHIFT=-175
CH7+DISPLAY_SHIFT=-175
CH7+JOYSTICK_SMOOTH=80
CH7+DISPLAY_AMPLIFY=5
CH7+BASELINE_SHIFT=225
CH7+SWITCH_SMOOTH=3
CH8+SWITCH_SHIFT=-175
CH8+DISPLAY_SMOOTH=77
CH8+JOYSTICK_AMPLIFY=5
CH8+SWITCH_AMPLIFY=5
CH8+JOYSTICK_SHIFT=-175
CH8+DISPLAY_SHIFT=-175
CH8+JOYSTICK_SMOOTH=80
CH8+DISPLAY_AMPLIFY=5
CH8+BASELINE_SHIFT=254
CH8+SWITCH_SMOOTH=3
```