Laboratorio di Automatica

Robot *Nomad 150*:

- Descrizione hardware e software

- Nozioni di base per l'utilizzo del Robot

Studente: Fabrizio Romanelli

Docente: Ing. Francesco Martinelli

1 Introduzione

1.1 Oggetto:

Scopo della relazione è descrivere il Robot *Nomad 150* e le librerie C/C++ necessarie per la relativa gestione e fornire le nozioni di base per utilizzare il Robot.

1.2 Hardware:

L'hardware oggetto della trattazione è il seguente:

- Robot *Nomad 150* (riportato in Figura 1);
- Radiomodem Mercury-EN;
- Joystick Mach 2.

Figura 1: Il Robot *Nomad 150*

1.3 Software:

Il software disponibile, è composto da librerie C/C++ per la gestione del Robot e dal programma Nserver (nelle versioni per Silicon Graphics: MIPS/IRIX5.3; SUN; LINUX).

1.4 Manuali:

I manuali[1] forniti con la strumentazione sono i seguenti:

- User's Manual
- Nomad 200 Hardware Manual
- Language Reference Manual

[1] I manuali si riferiscono al Robot *Nomad 200*, ma comprendono comunque tutte le informazioni necessarie all'utilizzo del Robot *Nomad 150*

2 Installazione e settaggio hardware e software

2.1 Installazione e settaggio hardware:

In questa sezione si vuole descrivere il procedimento di montaggio e settaggio dell'hardware.

Prima di mettere in funzione il Robot *Nomad 150* è necessario avere i seguenti accorgimenti:

- Dopo aver caricato le tre batterie del Robot (o dopo aver appurato che sono cariche), inserirle negli appositi alloggiamenti, dopo aver controllato che il joystick sia disinserito dalla porta *JOYSTICK* e avendo cura di rispettare l'ordine (su ognuna delle tre batterie c'è una lettera (a,b,c) alla quale ne corrisponde una sul Robot).
- Dopo aver controllato che nessuno dei connettori delle batterie appena inserite e nessun altro cavo/connettore sporga, disinserire il blocco del cilindro esterno, e far scorrere il cilindro fino alla base del Robot, facendo attenzione a non farlo toccare con i cavi ed i connettori interni al cilindro.
- Accendere il Robot tramite l'interruttore.

Per mettere in funzione il Robot occorre prima decidere il modo con il quale deve essere connesso al computer.
Vi sono tre diverse modalità di comando:

- Joystick
- Comandi elementari da programma emulazione terminale:
  - con cavo seriale su *serial* 1 interna al robot;
  - tramite radiomodem.
- Comandi complessi tramite programma *Nserver*:
  - con cavo seriale su *serial* 2 interna al robot;
  - tramite radiomodem.

La connessione tramite Joystick avviene nel modo seguente:
si collega il Joystick Mach 2 nella porta *JOYSTICK* del Robot. In questo modo il Robot è pronto a funzionare tramite Joystick.[2]

Una diversa modalità di comando è quella con programma di emulazione terminale ("minicom" sotto Linux o "Hyperterminal" sotto Windows) in questa modalità è possibile istruire il robot con comandi elementari. Esistono due possibilità di collegamento:

- connessione con cavo seriale su *serial* 1 interna al robot;
- connessione tramite radiomodem.

La connessione tramite cavo seriale su *CONSOLE* viene effettuata collegando il cavo seriale tra la porta seriale del computer e la porta *CONSOLE* del Robot, collegata a sua volta alla porta *serial* 1 interna. Si deve verificare che i collegamenti siano stati effettuati nel modo giusto, occorre quindi: aprire la parte superiore del Robot; controllare che la porta *CONSOLE* del Robot sia collegata alla porta *serial* 1 interna. Il collegamento è quindi effettuato.

La connessione tramite radiomodem viene effettuata collegando il radiomodem Mercury-EN alla porta seriale del computer tenendo presente che la porta seriale del PC deve essere impostata al valore di 9600 baud. Inoltre occorre agire direttamente sul Robot nel modo che segue: aprire la parte superiore del Robot; controllare che il radiomodem sia collegato direttamente alla porta *serial* 1 interna al Robot; in caso contrario collegare il radiomodem alla porta *serial* 1. Il collegamento è effettuato.

Una ulteriore modalità di comando è tramite programma *Nserver* (sotto Linux). Questo programma permette di far eseguire al robot comandi complessi, inoltre è possibile programmare il robot in C/C++ utilizzando il programma *Nserver* come programma *server* ed il programma scritto in C/C++ come programma *client*[3]. Esistono due possibilità di collegamento:

- connessione con cavo seriale su *serial* 2 interna al robot;
- connessione tramite radiomodem.

[2]Per l'utilizzo del Joystick si rimanda alla sezione 2.3 a pagina 7.
[3]Per il funzionamento del programma *Nserver* si rimanda alla sezione 3.4 a pagina 10.

La connessione tramite cavo seriale con programma *Nserver* viene effettuata collegando il cavo seriale tra la porta seriale del computer e la porta *HOST* del Robot. Si deve anche in questo caso verificare che i collegamenti siano stati effettuati nel modo giusto, occorre: controllare che la porta *HOST* del Robot sia collegata alla porta *serial* 2. Il collegamento è effettuato.
Con questo tipo di collegamento è possibile eseguire programmi piú complessi rispetto a quelli di emulazione terminale: ad esempio si possono eseguire programmi scritti in C/C++ sfruttando le librerie del Robot.

La connessione tramite radiomodem viene effettuata collegando il radiomodem alla porta seriale del computer come descritto precedentemente. Occorre inoltre: aprire la parte superiore del Robot; controllare che il radiomodem sia collegato direttamente alla porta *serial* 2 interna al Robot; in caso contrario collegare il radiomodem alla porta *serial* 2. Il collegamento è effettuato.

2.2 Installazione e settaggio software:

In questa sezione si descrive il procedimento di installazione e settaggio del software fornito con il Robot *Nomad 150*.

Il software (per Linux) fornito con il Robot è composto da:

- Librerie C/C++
- Programma *Nserver*

Per l'installazione delle librerie e del programma *Nserver* si deve procedere nel modo che segue: occorre decomprimere il file denominato *host-2.6.9-i486-redhat5.1-linux.tgz* nella directory di lavoro[4].

Dopo aver installato il software e le librerie, occorre apportare delle modifiche ad un file denominato *robot.setup*, nel quale sono contenuti i vari settaggi del Robot[5].

[4]Il file *host-2.6.9-i486-redhat5.1-linux.tgz* contiene librerie e software che risultano sicuramente compatibili con sistemi Linux con distribuzione *Mandrake* e *Red Hat*; per altre distribuzioni non è stata testata la compatibilità, ma esiste comunque un file *host-2.6.9-i486-unknown-linux.tgz* che può essere utilizzato per queste altre distribuzioni (non testato!).

[5]Per una descrizione dettagliata dei vari settaggi contenuti nel file *robot.setup* e *world.setup* si rimanda al manuale *User's Manual*.

La modifica che *necessariamente* deve essere apportata per il corretto funzionamento del Robot è la seguente: in *[robot]* controllare, ed eventualmente cambiare, la linea: *model = ...* con *model = n150*. Questa linea permette di comunicare al programma *Nserver* che il Robot sul quale si vuole lavorare è il *Nomad 150*.

2.3 Utilizzo del Joystick per testare il funzionamento del Robot:

Al *Nomad 150* può essere connesso un Joystick con due assi (fornito con il Robot). Questo può essere utile se si desidera testare il funzionamento di tutte le parti che compongono il Robot; inoltre come primo approccio utilizzare il Joystick è utile per capire il funzionamento del *Nomad 150*[6].

[6]Per l'uso del Joystick si rimanda al manuale *User's Manual*.

3 Considerazioni sull'hardware e sull'utilizzo del software

3.1 Descrizione dell'hardware:

Il *Nomad 150* è un Robot dotato di un sistema meccanico composto da tre ruote, che traslano contemporaneamente (controllate da un primo motore) e che ruotano contemporaneamente (controllate da un secondo motore) e da un motore che permette la rotazione della torretta.
Il *Nomad 150* si può preferibilmente rappresentare come un sistema mobile composto da tre assi (traslazionale, rotazionale per le ruote, rotazionale per la torretta): in questo modo risulta facilitata la lettura e la comprensione delle funzioni contenute nelle librerie fornite dal costruttore del Robot. Il Robot può raggiungere una velocità traslazionale di 20 inch[7]/sec mentre la velocità di rotazione delle ruote e della torretta non può superare i 45 gradi/sec. Il sistema sensoriale del *Nomad 150* è composto da due sistemi sensoriali: il sistema "tattile" ed il sistema "ad ultrasuoni".
Il primo è composto da 20 sensori indipendenti di pressione (*switch*), disposti su due corone con 10 sensori ognuna, in modo da coprire 360 gradi con una risoluzione di 18 gradi. Il sistema "ad ultrasuoni" è invece composto da 16 sonar che possono fornire la distanza nel *range* [17,255] inch. La risoluzione ottenibile con questo sistema sensoriale è dunque 22.5 gradi.
Inoltre il *Nomad 150* è dotato di encoder sulle tre ruote che permettono di conoscere la posizione relativa del Robot.

3.2 Considerazioni sull'hardware:

L'hardware descritto presenta alcune limitazioni: ad esempio i 16 sonar di cui è dotato il *Nomad 150* permettono di raggiungere una risoluzione di 22.5 gradi; questo significa che, ad esempio in una applicazione in cui il Robot da fermo deve cercare la via piú libera e ci si deve dirigere utilizzando i sonar, la scelta della direzione da seguire, in termini di angolo relativo, é un multiplo di 22.5 gradi, dovendo i 16 sonar coprire uno spazio di 360 gradi. Per ovviare a questa limitazione si può pensare di acquisire i dati dei sonar piú volte: ad esempio si possono acquisire i dati dei sonar nella posizione attuale (0 gradi) e

[7] 1 inch = 2.54 cm.

poi ruotare la torretta di 11.25 gradi ed acquisire di nuovo i dati dei sonar, si ottiene in questo modo una risoluzione pari a 11.25 gradi, come se il Robot fosse dotato di 32, anziché 16 sonar, che permette di scegliere tra le possibili 32 direzioni da seguire la migliore. Questo procedimento può essere applicato anche piú volte per ottenere una maggiore risoluzione; ovviamente in questo modo il tempo necessario alla computazione per la scelta della direzione da seguire aumenta, come se fossero presenti sul Robot un multiplo di 16 sonar.

Un altro fattore da tener presente quando si ha a che fare con il *Nomad 150* è l'imprecisione degli encoder situati nelle ruote; infatti le ruote possono essere soggette a slittamenti sia dovuti alle imperfezioni del pavimento, sia alla possibilità che il Robot si trovi in prossimità di ostacoli: in questi casi infatti si presenta la possibilità che le misure effettuate sugli encoder diano risultati erronei: occorre quindi prestare particolare attenzione affinché queste condizioni non si verifichino e nel caso si verificassero, ad escludere dall'elaborazione i dati provenienti da queste misure non esatte.

Infine si deve tener presente che:la variazione dello *steering angle* non cambia la configurazione dei sensori, mentre la variazione del *turret angle* porta una rotazione della torretta e quindi dei sensori solidali con essa; l'unica eccezione è rappresentata dai bumper, che non cambiano mai la loro configurazione.

3.3 Descrizione del software:

Il software fornito con il *Nomad 150*, come già detto, è composto da:

- Librerie C/C++
- Programma *Nserver*

Le librerie C/C++ per il *Nomad 150* contengono le funzioni C/C++ necessarie per la gestione del Robot.
Nserver è un programma che genera una interfaccia grafica elaborata, che permette di vedere i dati provenienti dai sensori, la posizione del Robot e la mappa dello spazio in cui il Robot si muove; inoltre *Nserver* ha la capacità di funzionare anche da simulatore: questo può essere particolarmente utile se si vogliono testare i programmi senza utilizzare il Robot vero e proprio.

3.4 Considerazioni sul software:

Il Robot può funzionare in due modi:

- da un programma di emulazione terminale (*hyperterminal* sotto Windows e *minicom* sotto Linux), per far eseguire al Robot comandi elementari;
- usando il programma *Nserver* (sotto Linux) per far funzionare i programmi in C/C++ (eseguibili anche direttamente).

Entrambe le modalità sono accessibili con connessione diretta seriale oppure tramite radiomodem.
Il programma *Nserver* consente, come già detto, di utilizzare un simulatore per il Robot, molto utile nella fase di testing dei programmi[8]. Occorre fare delle considerazioni sul modo in cui devono essere compilati i programmi in C/C++ e sul modo in cui possono essere fatti girare sul computer.
I programmi C/C++ infatti possono essere compilati, ed eseguiti, in due modi: il primo modo è il collegamento tramite programma *Nserver*, che permette sia di testare il programma *Client* scritto in C/C++ con il simulatore, sia di eseguire il programma sul Robot reale; il secondo modo è quello di eseguire il programma direttamente sul Robot,

[8]Per l'utilizzo del programma *Nserver* si rimanda al manuale *User's Manual*.

senza avere in esecuzione il programma *Nserver*.
Nel primo caso occorre compilare il programma C/C++ utilizzando in Linux ad esempio la seguente sintassi:

gcc -o prova prova.c Nclient.o

Si specifica cioè il file *object* al quale deve fare riferimento il programma; in questo caso il file *object* è *Nclient.o*, dato che il programma C/C++ scritto deve comportarsi da *Client* rispetto al programma *Nserver* che si comporta da *Server*.
Nel secondo caso invece un esempio di compilazione potrebbe essere il seguente:

gcc -o prova prova.c Ndirect.o

Si specifica in questo caso il file *object* al quale deve far riferimento il programma; qui il file *object* è *Ndirect.o*, dato che il programma C/C++ deve collegarsi direttamente al Robot senza bisogno di programmi intermedi.
Il primo modo è il piú conveniente se si desidera testare il programma, prima con il Robot simulato, poi con il Robot reale, tenendo presente che *Nserver* deve essere in esecuzione prima di far partire il programma C/C++ scritto; nel caso il programma scritto funzioni in maniera corretta è quindi consigliabile utilizzare il collegamento diretto, senza *Nserver*, poiché in questo modo si minimizza la comunicazione tra Robot e computer.

4 Funzioni per l'utilizzo e la gestione del *Nomad 150*

4.1 Oggetto:

In questa sezione vengono descritte alcune delle procedure piú importanti per utilizzare il *Nomad 150*. Le funzioni descritte nel seguito si riferiscono a funzioni C fornite nelle librerie messe a disposizione dal costruttore del Robot[9].

4.2 Descrizione dei vettori globali:

Le informazioni sullo stato corrente del Robot, la sua configurazione e le letture dei sensori possono essere ottenute da un "array globale", chiamato *State Vector*. Questo vettore è aggiornato dopo l'esecuzione di un determinato comando del Robot; i campi dello *State Vector* che devono essere aggiornati, sono contenuti nel vettore globale *Smask* (State Mask).
I nomi dei campi sono valori definiti nel file *Nclient.h*; è preferibile utilizzare questi valori, come ad esempio *STATE_VEL_TRANS* per la velocità traslazionale del Robot, piuttosto che l'indice nell'array, incrementando in questo modo la leggibilità del codice[10].

4.3 Inizializzazione del Robot:

Per poter utilizzare il *Nomad 150* occorre inizializzarlo. La procedura di inizializzazione consta delle seguenti fasi:

- connessione al Robot;
- configurazione *timeout* del Robot;
- azzeramento del Robot;
- assegnazione dei parametri di accelerazione e velocità.

La connessione al Robot viene effettuata tramite la funzione seguente:

connect_robot(robot_id)[11]

[9] Per il loro impiego specifico si rimanda al manuale *Language Reference Manual*.
[10] Si rimanda al manuale *User's Manual* per la lista completa dei valori dei vettori globali descritti in questa sezione.
[11] Per disconnettere il Robot, prima che il programma *Client* termini, occorre chiamare la funzione *disconnect_robot(robot_id)* in maniera analoga alla funzione *connect_robot(robot_id)*.

Questa funzione è necessaria per richiedere una connessione al Robot tramite Server, essendo *robot_id* l'identificativo del Robot. Il valore di *default* per la variabile *robot_id* è 1. Se la connessione al Server va a buon fine, la funzione *connect_robot( )* restituisce *robot_id*.

La configurazione del *timeout* del Robot si esegue richiamando la funzione seguente:

conf_tm(timeout)

Questa funzione setta il timeout del Robot a *timeout* secondi: in questo modo se il Robot non riceve alcun comando dall'*host* per un tempo maggiore del periodo di timeout, termina il movimento corrente; questa funzione è una misura di sicurezza per prevenire che il Robot continui il suo moto in maniera incontrollata, se, per qualche ragione, non riceve alcun comando dall'*host*.

L'azzeramento del Robot è necessario per allineare le ruote e la torretta con lo zero dei bumper. La funzione addetta a tale compito è:

zr( )

Questa funzione allinea automaticamente la direzione anteriore delle ruote e della torretta con la direzione anteriore dei bumper e setta gli angoli delle ruote e della torretta a 0, ponendo a (0,0) anche le coordinate (x,y) della posizione del Robot.
La funzione restituisce il valore 1 in caso di operazione riuscita con successo, mentre restituisce 0 in caso contrario.

L'assegnazione dei parametri di accelerazione e velocità si effettua chiamando le seguenti funzioni:

ac(t_ac,s_ac,r_ac)

sp(tsp,ssp,rsp)

La prima funzione assegna le accelerazioni traslazionali, di rotazione delle ruote e di rotazione della torretta del Robot. Le accelerazioni sono date in 1/10 inch/sec^2 per l'accelerazione traslazionale e in 1/10 grado/sec^2 per le accelerazioni di rotazione delle ruote e della torretta. I parametri da passare alla funzione sono:

- *t_ac* per l'accelerazione di traslazione;
- *s_ac* per l'accelerazione di rotazione delle ruote;
- *r_ac* per l'accelerazione di rotazione della torretta.

La funzione restituisce 1 in caso di successo, 0 altrimenti.

La seconda funzione assegna le velocità di traslazione, rotazione delle ruote e della torretta del Robot.Le velocità sono date in 1/10 inch/sec per la velocità traslazionale nel *range* [0,200], ed in 1/10 gradi/sec per le velocità di rotazione delle ruote e della torretta nel *range* [0,450]. I parametri da passare alla funzione sono:

- *tsp* per la velocità di traslazione;
- *ssp* per la velocità di rotazione delle ruote;
- *rsp* per la velocità di rotazione della torretta.

La funzione restituisce 1 in caso di successo, 0 altrimenti.

4.4 Istruzioni per il movimento del Robot:

Esistono diverse funzioni che istruiscono il Robot sui movimenti che deve eseguire:
la prima funzione è la seguente:

pr(tpr,spr,rpr)

Questa funzione permette al Robot di muoversi per una determinata distanza, e ruotare, usando come velocità quelle settate con la funzione *sp(,,)* al momento dell'inizializzazione. I parametri da passare alla funzione *pr(,,)* sono i seguenti:

- *tpr* per la distanza che deve essere percorsa, in 1/10 di inch, nel *range* [-32000,32000];
- *spr* per l'angolo di cui devono ruotare le ruote del Robot, in 1/10 di gradi, nel *range* [-32000,32000];
- *rpr* per l'angolo di cui deve ruotare la torretta del Robot, in 1/10 di gradi, nel *range* [-32000,32000].

La funzione restituisce 1 in caso di successo, 0 altrimenti.

Esiste una seconda funzione:

mv(t_mode,t_mv,s_mode,s_mv,r_mode,r_mv)

Questa funzione permette al programmatore di far eseguire un movimento indipendente dei tre assi del Robot, utilizzando per ogni asse una determinata legge di controllo.
Per ogni asse vi sono due argomenti:

1. Il primo argomento determina la legge di controllo specificando:
   - *MV_VM* per il controllo in velocità,
   - *MV_PR* per il controllo in posizione,
   - *MV_IGNORE* se il movimento per l'asse corrente deve rimanere inalterato.
2. Il secondo argomento rappresenta una velocità, se è stato specificato il controllo in velocità oppure una posizione, se è stato specificato il controllo in posizione.

La funzione restituisce *TRUE* se il comando è stato eseguito con successo, mentre restituisce *FALSE* in caso contrario o se gli argomenti inseriti non sono corretti.

Lo scopo della seguente funzione è quello di far eseguire un movimento al Robot in accordo alle velocità specificate dai suoi parametri:

vm(tv,sv,rv)

Gli argomenti da passare alla funzione sono i seguenti:

- *tv* è la velocità traslazionale desiderata in 1/10 di inch/sec, nel *range* [-240,240];
- *sv* è la velocità di rotazione delle ruote desiderata in 1/10 di gradi/sec, nel *range* [-450,450];
- *rv* è la velocità di rotazione della torretta desiderata in 1/10 di gradi/sec, nel *range* [-450,450].

La funzione restituisce 1 in caso di successo, 0 altrimenti.

La seguente funzione fa fermare il Robot:

st( )

Con questo comando il Robot viene fermato in modo *controllato* con accelerazioni appropriate, mantenendo la sua posizione attuale. Da notare che se l'accelerazione è 0, il Robot NON si ferma.
La funzione restituisce 1 in caso di successo, 0 altrimenti.

Esiste un'altra funzione per fermare il Robot:

ws(w_t,w_s,w_r,timeout)

Con questo comando si attende che i motori del Robot siano fermi.
I parametri da passare alla funzione sono:

- *w_t* specifica se attendere che l'asse di traslazione si sia fermato: il valore 1 indica che si deve attendere, 0 che non si deve attendere;
- *w_s* specifica se attendere che l'asse di rotazione delle ruote si sia fermato: 1 indica che si deve attendere, 0 che non si deve attendere;
- *w_r* specifica se attendere che l'asse di rotazione della torretta si sia fermato: 1 indica che si deve attendere, 0 che non si deve attendere;
- *timeout* indica che se non tutti gli assi specificati si sono fermati prima che siano trascorsi *timeout* secondi, la funzione termina.

Questa funzione restituisce 1 in caso di successo, 0 altrimenti.

4.5 Istruzioni per la configurazione dei sensori e per l'acquisizione dei dati:

In questa sezione si descrivono quei comandi che permettono al programmatore di configurare i sensori e di acquisire i dati da essi.
Per aggiornare i valori dello *State Vector* in accordo con lo *Smask Vector*, e quindi tutti i parametri che definiscono lo stato del Robot, occorre chiamare la funzione:

gs( )

Questa restituisce 1 in caso di successo e 0 altrimenti. È molto utile chiamare spesso questa funzione se si vuole che i valori dello *State Vector* siano aggiornati in tempo reale: si confronti a titolo di esempio il seguente frammento di codice:

```
vm(50,0,0); // si muove di 5 inch/sec in avanti
// nella seguente linea di codice si controlla continuamente che
// la condizione (State[STATE_SONAR_0] > 10) sia verificata,
// aggiornando ad ogni ciclo lo State Vector
while( State[STATE_SONAR_0] > 10)
        gs(); // aggiorna lo State Vector
st(); // ferma il Robot
ws(1,0,0,1); // attende che il Robot sia fermo
```

Per aggiornare solo i dati dei sonar nello *State Vector* bisogna richiamare la funzione:

get_sn( )

Questa acquisisce le informazioni provenienti dai sonar, quelli attivati per mezzo della funzione *conf_sn(,)*, ed aggiorna gli indici dello *State Vector* corrispondenti ai valori dei 16 sonar.

Per aggiornare solo i dati dei *bumper*, sensori di pressione, nello *State Vector* occorre richiamare la seguente funzione:

get_bp( )

Questa acquisisce le informazioni provenienti dai bumper ed aggiorna gli indici dello *State Vector* corrispondenti ai valori dei 20 bumper.
Vale la pena di soffermarsi sulla lettura del valore di *State[STATE_BUMPER]*. Il bumper numero n è rappresentato dall'$(n+1)$-esimo bit del valore contenuto in *State[STATE_BUMPER]*. Quando un bumper viene colpito (il Robot tocca un ostacolo), il bit corrispondente a quel bumper viene settato ad 1. Di seguito si riporta un frammento di codice in cui viene rappresentato un esempio di lettura del contenuto di *State[STATE_BUMPER]*:

```
for(j=0;j<20;j++) // si ripete per tutti i 20 bumper
// la seguente condizione dà TRUE se il bumper j è toccato
// dà FALSE se non è toccato, cioè esegue l'AND tra lo
// State[STATE_BUMPER] ed il vettore 1L (cioè 1 in formato
// Long, 32 bit), shiftando ad ogni passo del for di j bit
// verso sinistra
  if(State[STATE_BUMPER] & ( 1L << j))
  {
   printf(1 ); // scrive 1 se il bumper j è stato toccato
  }
  else
   printf(0 ); // scrive 0 se il bumper j non è stato toccato
```

Per ottenere i dati sulla configurazione del Robot si utilizza la seguente funzione:

get_rc( )

Questa aggiorna i seguenti indici dello *State Vector*: *STATE_CONF_X*, *STATE_CONF_Y*, *STATE_CONF_STEER*, *STATE_CONF_TURRET*.

Si riporta infine una funzione che permette di configurare i sonar:

conf_sn(rate,order[16])

I parametri che le devono essere passati sono i seguenti:

- *rate* specifica il rate di emissione dei sonar a intervalli di tempo di 4 milli-secondi, il parametro *rate* deve essere settato ad un valore compreso tra 0 e 255;

- *order[16]* specifica la sequenza di emissione dei sonar. Il sonar specificato in *order[i]* emetterà prima del sonar specificato in *order[i+1]*.

Da notare che i sonar non specificati in *order[ ]* non vengono attivati.

4.6 Istruzioni per il programma Server:

I comandi introdotti in questa sezione richiedono che il programma *client* sia connesso al programma *Nserver*. Le chiamate dai programmi connessi direttamente al Robot (cioè linkati con *Ndirect.o*) non produrranno alcun effetto.
La funzione:

server_is_running( )

restituisce il valore *TRUE* se *Nserver* ed il programma *Client* sono connessi allo stesso socket.

La funzione:

quit_server( )

se chiamata da un programma *Client* connesso ad *Nserver*, chiude il programma *Nserver*.

Di seguito è riportata una funzione che permette di aggiungere un ostacolo alla rappresentazione del mondo del programma server:

add_obstacle(obs[21])

L'argomento da passare a questa funzione è l'array *obs[21]* nel quale:
obs[0] specifica il numero di vertici dell'ostacolo poligonale (massimo 10 vertici); *obs[2i+1]* e *obs[2i+2]* specificano le coordinate x ed y del vertice i-esimo del poligono.

Per cancellare un ostacolo già inserito nella rappresentazione del mondo, occorre utilizzare la seguente funzione:

delete_obstacle(obs[21])

passandole per parametro l'array *obs[21]* come definito precedentemente[12].

Si riportano infine due comandi per passare dal Robot reale al Robot simulato:

real_robot( )
simulated_robot( )

La funzione *real_robot( )*, chiamata da un programma *client*, fa in modo che il *Server* invii i comandi dall'applicazione *Client* al Robot reale piuttosto che inviarli al Robot simulato. La funzione *simulated_robot( )* ha l'effetto inverso della precedente.

[12]Per ulteriori comandi per il disegno della rappresentazione del mondo, si rimanda al manuale *Language Reference Manual*.

5 Considerazioni finali

5.1 Conclusioni:

Scopo della relazione è quello di fornire una conoscenza di base del software e dell'hardware per il Robot *Nomad 150*. Si è cercato di mettere in evidenza nella presente trattazione le caratteristiche dell'hardware e del software con particolare attenzione al relativo settaggio ed installazione.
Si è ritenuto importante inoltre soffermarsi sulla descrizione della strumentazione e sulle relative limitazioni.

La parte principale della relazione è stata dedicata alla programmazione, descrivendo alcune delle funzioni C/C++ messe a disposizione dal costruttore del Robot. Partendo da queste semplici funzioni è possibile creare programmi complessi e gestire il *Nomad 150* in modo avanzato.

L'hardware descritto si è dimostrato particolarmente versatile; questo ha permesso di capirne facilmente il funzionamento e le caratteristiche. Il *Nomad 150* risulta essere un valido strumento che può essere impiegato con buoni risultati in semplici applicazioni di navigazione robotica.

Per incrementare le prestazioni del Robot, e quindi estendere il campo delle applicazioni in cui può essere utilizzato, si può pensare di integrare al sistema *Robot + Computer* una telecamera per acquisire immagini da elaborare e da utilizzare nella navigazione del Robot.

6 Appendice

6.1 Esempio di programma C per il *Nomad 150*:

```
// File di esempio

#include<stdlib.h>
#include<stdio.h>
#include<math.h>
#include"Nclient.h"

#define COEFF_DIST 0.254 // conversione da decimi di inch a centimetri
#define COEFF_ANGOLO 0.0017453 /* conversione da decimi di grado a
                                  radianti */

void CREA_MAPPA ()
{
  // specifica i vertici di un rettangolo
  long rettangolo[21]={4,200,-100,600,-100,600,100,200,100};

  sleep(1); // attende 1 secondo prima di eseguire la prossima istruzione
  printf("Sto creando la mappa...\n");

  new_world(); // cancella tutti gli ostacoli dall'ambiente corrente
  add_obstacle(rettangolo); // crea il rettangolo specificato
  move_obstacle(rettangolo,600,0); // muove il rettangolo specificato
  add_obstacle(rettangolo);
  move_obstacle(rettangolo,0,1000);
  add_obstacle(rettangolo);
  move_obstacle(rettangolo,0,-2000);
  add_obstacle(rettangolo);
  move_obstacle(rettangolo,-1800,0);
  add_obstacle(rettangolo);
```

```
  move_obstacle(rettangolo,1000,0);
  add_obstacle(rettangolo);
  move_obstacle(rettangolo,0,2000);
  add_obstacle(rettangolo);
  move_obstacle(rettangolo,-1000,0);
  add_obstacle(rettangolo);
  move_obstacle(rettangolo,0,-1000);

  printf(mappa creata...\n);
}

// la seguente funzione legge la posizione del robot

void LEGGI_ENCODER(double *Stato)
{
  get_rc();
  // converte da decimi di grado a radianti
  Stato[0] = State[STATE_CONF_STEER]*COEFF_ANGOLO;
  // converte le X da decimi di pollici a centimetri
  Stato[1] = State[STATE_CONF_X]*COEFF_DIST;
  // converte le Y da decimi di pollici a centimetri
  Stato[2] = State[STATE_CONF_Y]*COEFF_DIST;
}
```

```
// la seguente funzione attende l'inizio e la conclusione del movimento

void ATTENDI(int asse)
{
  if(State[asse] == 0) // attende che abbia inizio il movimento
    gs();

  while(State[asse] != 0) // attende che sia completato il movimento
    gs();
}

// la seguente funzione restituisce il valore minimo dei 3 sonar frontali

int MIN_DISTANZA()
{
  int i,min = 255;

  // seleziona i 3 sonar frontali
  for(i=0;i<2;i++)
    if(State[STATE_SONAR_0 + i] < min)
      min = State[STATE_SONAR_0 + i];
  for(i=14;i<15;i++)
    if(State[STATE_SONAR_0 + i] < min)
      min = State[STATE_SONAR_0 + i];

  return(min); // restituisce la minima distanza letta dai sonar
}
```

```
// la seguente funzione è addetta alla scelta della direzione
// migliore da seguire; restituisce l'angolo di cui deve ruotare
// il robot per non incontrare ostacoli

int SCELTA_DIREZIONE(int angolo_prec)
{
  int i,dir,max = 0;

  // si usano tutti i sonar
  for(i = STATE_SONAR_0;i<=STATE_SONAR_15;i++)
    if((State[i]>max)&&(angolo_prec!=(dir=i - STATE_SONAR_0) * 225))
        max = State[i];
  // sceglie la rotazione a destra o a sinistra...
  if(dir > 8)
    dir = dir - 16;

  // data la soglia di 20 inch, sceglie la direzione migliore
  if(max > 20)
    /* si moltiplica per 225 decimi di grado, cioè 360 gradi/16 sonar=22,5
       gradi */
    return(dir * 225);
  else
    return(-1);
}

int main (void)
{
  int angolo,angolo_prec,min,k = 21;
  double Stato[3];

  if(!connect_robot(1)) // se la connessione al Robot fallisce, esce
```

```
    exit(0);

  conf_tm(2); // setta il timeout del robot in secondi
  zr(); // allinea le ruote e la torretta a zero
  ac(200,300,300); // setta le accelerazioni
  sp(100,200,200); // setta le velocità

  CREA_MAPPA(); // crea la mappa

  init_sensors(); // inizializza i sensori
        printf(Inizio la navigazione...\n);

// il seguente ciclo while permette di selezionare l'angolo libero,
// se ce n'è uno, e di far muovere il Robot nella direzione
// individuata dal detto angolo
  while( (angolo = SCELTA_DIREZIONE(angolo_prec)) != -1)
    {
          pr(0,angolo,angolo); // si gira verso l'angolo libero
          ATTENDI(STATE_VEL_STEER);
          vm(50,0,0); // si muove a 5 inch/sec nella direzione scelta
          // nel seguente while si controlla continuamente che la
          // distanza minima non scenda sotto i 10 inch
          while( (min = MIN_DISTANZA()) > 10)
                    gs();
          st(); // ferma i motori
          ws(1,1,1,1); // attende che tutti i motori si siano fermati
          angolo_prec = angolo;

          // esce dal ciclo while se la distanza si fa minore di 8 inch
          if( min <= 8 ) break;
    }
```

```
  printf(Premere un tasto per acquisire i dati dai sensori...\n);
  getchar();

  printf(Dati sonar:  );
  int l,j;
  for(l=0;l<16;l++)
    printf(%d , State[STATE_SONAR_0+l]);

  printf(\nDati bumper:  );
  for(j=0;j<20;j++)
    if(State[STATE_BUMPER] & ( 1L << j))
      {
        printf(1 );
        k = j;
      }
    else
      printf(0 );

  printf(\n);

  if(k != 21)
  {
    printf(\nIl bumper toccato e' il numero %d...\n,k);
    printf(Il bumper e' riferito alla configurazione iniziale\n);
  }

  LEGGI_ENCODER(Stato);
  printf(Steer = %f\n,Stato[0]); // in radianti
  printf(X = %f\n,Stato[1]); // in centimetri
  printf(Y = %f\n,Stato[2]); // in centimetri
  disconnect_robot(1); // si disconnette dal Robot
}
```

Indice