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

Laboratorio di Robotica Avanzata
Advanced Robotics Laboratory

Corso di Robotica Mobile
(Prof. Riccardo Cassinis)

Ripristino AMIROLOS marker

Elaborato di esame di:

Beniamino Galvani, Giovanni Maggini, Claudio Marranzino Scopece

Consegnato il: **23 Luglio 2007**

Sommario

L'elaborato svolto si integra in un più ampio progetto che si propone di sviluppare un sistema di posizionamento a basso costo per il robot ActivMedia Pioneer 3 "MORGUL". Al fine di identificare la posizione del robot viene utilizzato un gruppo di led ad alta intensità che lampeggiano ad una determinata frequenza scelta dall'utente. Elaborando quindi le immagini riprese da una webcam è possibile riconoscere i lampeggi dei led e quindi la posizione del robot. Avendo a disposizione un PIC per il controllo del gruppo di led e motore (marker), nasce la necessità di avere delle librerie a disposizione per interfacciare il programma ARIA al firmware del PIC. Scopo di questo lavoro è, quindi, la rimessa in funzione del marker e la documentazione delle librerie software per il funzionamento, mettendo così a disposizione degli studenti un documento completo contenente tutta la documentazione esistente riguardante il marker del robot Morgul.

1. Introduzione

Questo progetto è nato dalla necessità di dover aggiornare e ripristinare il marker installato sul robot Morgul. Il marker sostanzialmente altro non è che un dispositivo collegato al robot costituito da un motore, una fonte luminosa ed uno specchio particolare. Il motore è essenziale per la diffusione della luce dove opportuno, in quanto permette di rendere direzionale la fonte luminosa e quindi oltre che a permettere la localizzazione a 360 gradi del robot da parte delle videocamere, anche di risparmiare energia; cosa utile, dato che il robot è dotato di proprie batterie che vanno sfruttate al meglio sia per portare a termine operazioni essenziali al suo funzionamento sia per la realizzazione dell'obiettivo. La fonte luminosa è realizzata mediante un insieme di led che nel caso specifico sono di un'unica tonalità di colore, ma con possibilità di variare gli impulsi luminosi in durata e frequenza. In aggiunta è presente la predisposizione per due gruppi di led addizionali. Lo specchio inclinato di circa 45° posto in direzione dei led è utile per catturare la luce e proiettarla dove è di interesse.

Il marker di questo caso di studio costituisce parte del modulo di localizzazione AMIRoLoS (Active Marker Internet-based Robot Localization System) del progetto SAURON (Surveillance AUtonomous Robots Over Network), dedicato alla progettazione di robot autonomi a scopi di sorveglianza.

SAURON prevede la creazione di una rete per rilevare la posizione del robot all'interno di un'area all'interno del raggio di visuale di videocamere, gestibile tramite un'interfaccia web. Il marker attivo montato su Morgul permette la localizzazione del robot da parte di una webcam dedicata al riconoscimento del pattern eseguito dalla luce dei led ed è stato studiato appositamente per essere immune alle interferenze dovute alla luce solare in spazi aperti. Al controllo del marker è deputato un microcontrollore PIC 16F876, montato su una scheda logica collegata al calcolatore utilizzato per il controllo del robot Morgul. [1]

Il lavoro che è stato compiuto è di miglioramento dei lavori portati a termine negli anni precedenti, che necessitano di manutenzione e aggiornamento col passare del tempo.

Per il controllo del marker del robot Morgul prima di tutto ci si è dovuti documentare e quindi si sono analizzate le librerie già esistenti, operazione necessaria per poter effettuare eventuali modifiche.

Dopo aver avuto una generica ma completa visione su ciò che i diversi metodi e funzioni principali andavano ad applicare ci si è accorti di alcune parti di codice non perfettamente funzionanti o da completare, ed è su queste che si è concentrata la nostra attenzione durante lo svolgimento del lavoro.

1.1. Il robot Morgul

Il robot utilizzato (nome in codice Morgul: Mobile Observation Robot for Guarding the University Laboratories) è un modello Pioneer 3 AT della ActivMedia, basato sul microprocessore Hitachi H8S. Pioneer è una famiglia di robot mobili a due e quattro ruote motrici sviluppati da ActivMedia come piattaforme di ricerca e sviluppo, con modelli che includono il Pioneer 1 e il Pioneer AT, passando per le varie versioni del Pioneer 2 per arrivare poi a uno dei modelli più recenti, il Pioneer 3, che è appunto quello che abbiamo avuto a disposizione per questo progetto. Tutti i robot della serie Pioneer condividono la medesima architettura di controllo client-server di tutti gli altri modelli ActivMedia. [2]

Morgul è dotato, dal punto di vista software, di un sistema operativo installato nel controller del robot chiamato AROS (ActivMedia Robotics Operating System) e di un host per il supporto di applicazioni client avanzate di controllo del robot e per ambienti di sviluppo per applicazioni. Per lo sviluppo software ci si può avvalere delle librerie di ActivMedia chiamate ARIA (ActivMedia Robotics Interface for Applications) rilasciate sotto licenza GNU Public License, del sistema di sviluppo Saphira di SRI International (ormai obsoleto) oppure di software rilasciati da terze parti.

Dal punto di vista tecnico, Morgul ha una struttura di alluminio di dimensioni 50cm x 49cm x 26cm con ruote dal diametro di 21.5 centimetri, ognuna delle quali è mossa da un motore con un rapporto di marcia 66:1 contenente encoder da 100 passi, utilizzati per controlli odometrici sulla posizione. Le quattro ruote creano una struttura skid-steer anolonoma, ovvero il robot Morgul è capace di ruotare su se stesso muovendo i due gruppi di ruote (destro e sinistro) con senso di rotazione opposto oppure può muovere solo le ruote di un singolo lato, formando un cerchio di 40cm di raggio. [3]

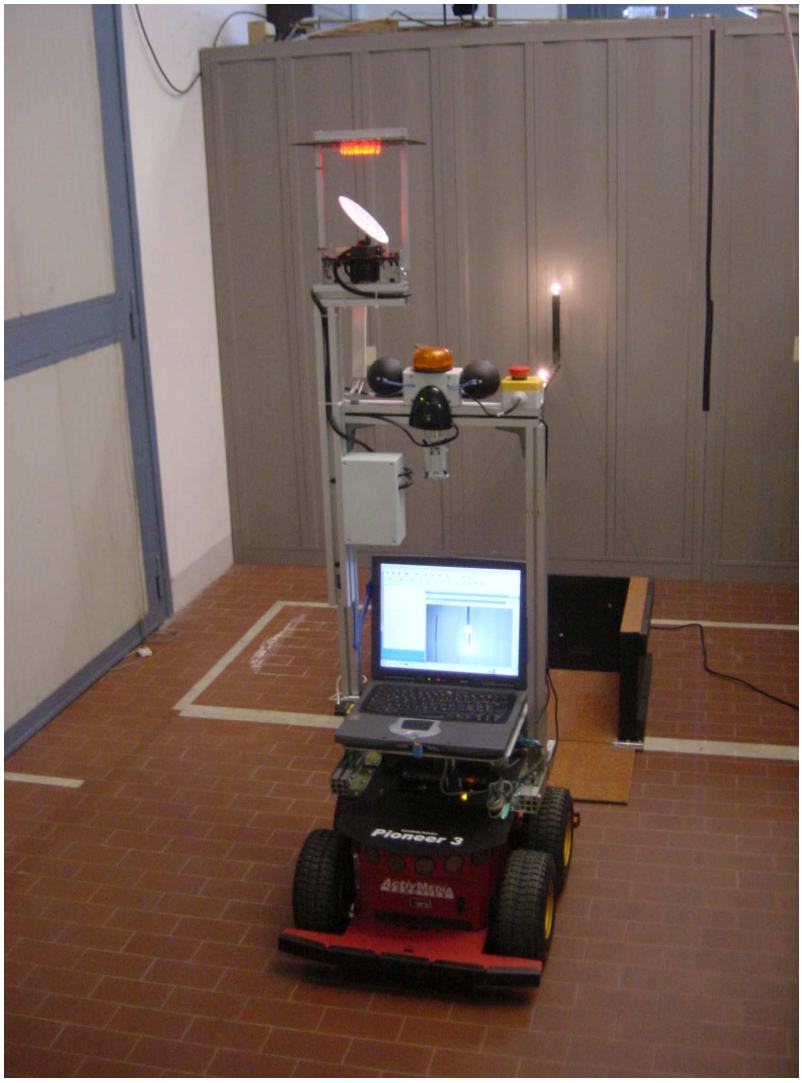

Figura 1 : Il robot Morgul nel Laboratorio di Robotica Avanzata dell'Università

Il marker è collocato sulla torretta del robot, protetto dalla luce solare da una copertura in plastica nera per ridurre riflessi e rendere meglio visibile la luce dei led, come è possibile osservare dalla Figura 2.

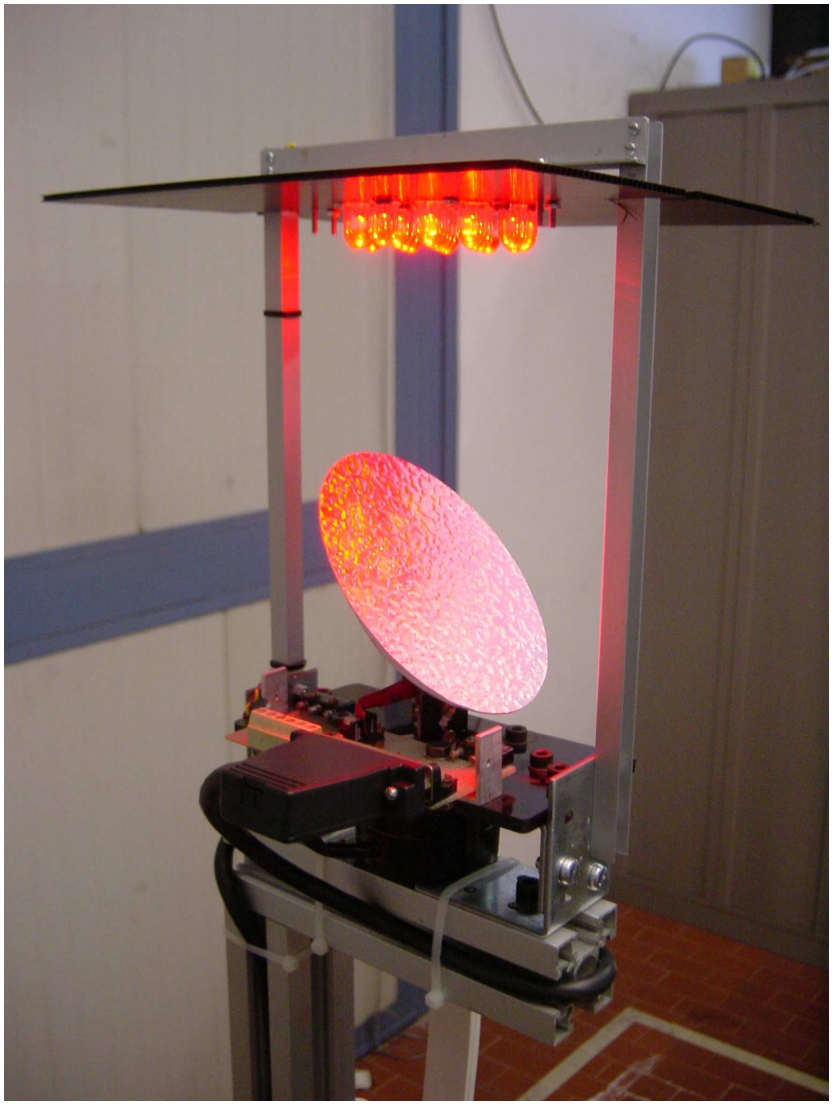

Figura 2 : Particolare del robot Morgul (torretta rotante con specchio del marker)

Il software usato per questo progetto è basato sulle librerie ARIA. ARIA, acronimo di ActivMedia Robotics Interface for Applications, è un ambiente di sviluppo open-source basato sul C++ che fornisce un'interfaccia lato-client a una varietà di sistemi robotici, tra i quali il controller del Pioneer 3. [4] ARIA gestisce le interazioni client-server anche di basso livello, quali le comunicazioni seriali, il processing dei pacchetti di comando e di informazioni server, il multithreading e permette di controllare vari tipi di apparecchi quali sensori, giroscopi, fotocamere e altri.

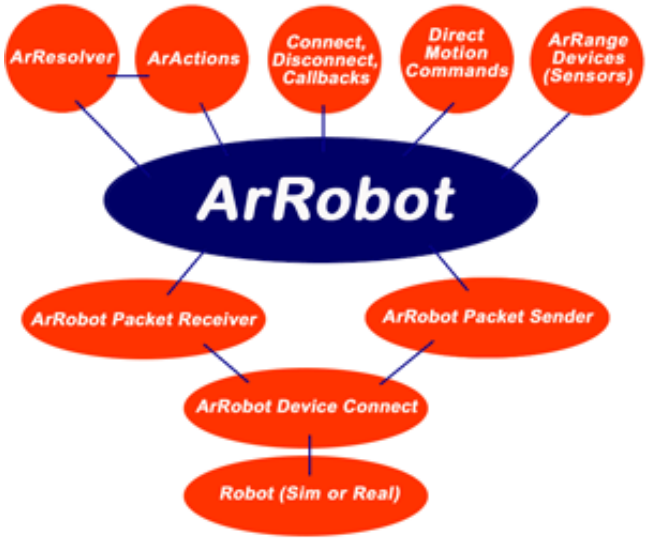

Figura 3 : Architettura di ARIA

1.2. Il gruppo di lavoro

Il gruppo di lavoro ha partecipato con interesse alla revisione delle funzionalità del marker dopo che il professore ha proposto il lavoro ed ha dedicato anche qualche lezione per la spiegazione del funzionamento dello stesso e dei suoi problemi. Il gruppo è comunque costituito da 3 studenti che hanno seguito il corso di robotica mobile nell'anno corrente e lo hanno trovato interessante.

La maggior parte del lavoro si è svolta a uno dei calcolatori messi a disposizione nel Laboratorio di Robotica Avanzata (ARL) dell'Università, collegandosi tramite protocollo SSH e rete wireless al calcolatore installato sul robot Morgul. Abbiamo anche potuto lavorare da casa, collegandoci in remoto al robot permettendoci di effettuare modifiche e testare il programma.

1.3. Struttura dell'elaborato

L'elaborato è suddiviso in diversi paragrafi: un'introduzione, seguita dall'esposizione del problema affrontato dal nostro gruppo di lavoro e la soluzione adottata per risolverlo. In seguito ci occuperemo di documentare quanto è stato fatto in modo da fornire un esaustivo documento di HOWTO per futuri utilizzi.

2. Il problema affrontato

Nel momento in cui abbiamo iniziato a lavorare a questo progetto, avevamo a disposizione solo un programma di test del marker non funzionante (non eseguiva la operazioni desiderate e si bloccava durante l'esecuzione) e una documentazione del software del PIC che controlla il marker.

Il programma in questione (il cui nome del file è testmarker) si basava su librerie realizzate per il software ARIA che creavano un'interfaccia tra l'hardware e i comandi inviati tramite porta seriale. Queste librerie (realizzate da Marcello Oberti) non erano documentate e il lavoro svolto sul PIC da Manzini si basavano su tipi diversi di librerie software.

Il problema affrontato è stato quindi quello di studiare le librerie su cui era basato il programma di test messoci a disposizione, studiare il programma di controllo del PIC e modificare le librerie preesistenti in modo da eliminare i difetti che ne impedivano il corretto funzionamento.

La scheda di controllo del marker è basata sul microcontrollore PIC 16F876 prodotto dalla Microchip che comunica con il robot attraverso una connessione di tipo seriale realizzata con un integrato MAX232 in grado di convertire i livelli di tensione. Lo specchio è posizionato utilizzando un motore a passo (200 passi da 1,8 gradi ciascuno) comandato dalla scheda driver Toshiba TA8435. Il PIC è stato programmato con un firmware che consente il controllo da parte dell'utente del motore a passo e del lampeggio dei led utilizzando comandi inviati sotto forma di stringhe di testo (anche i numeri sono inviati sotto forma di caratteri) sfruttando la connessione seriale. Allo stesso modo, il PIC è in grado di inviare segnali, sempre sotto forma di caratteri, in risposta a determinati eventi.

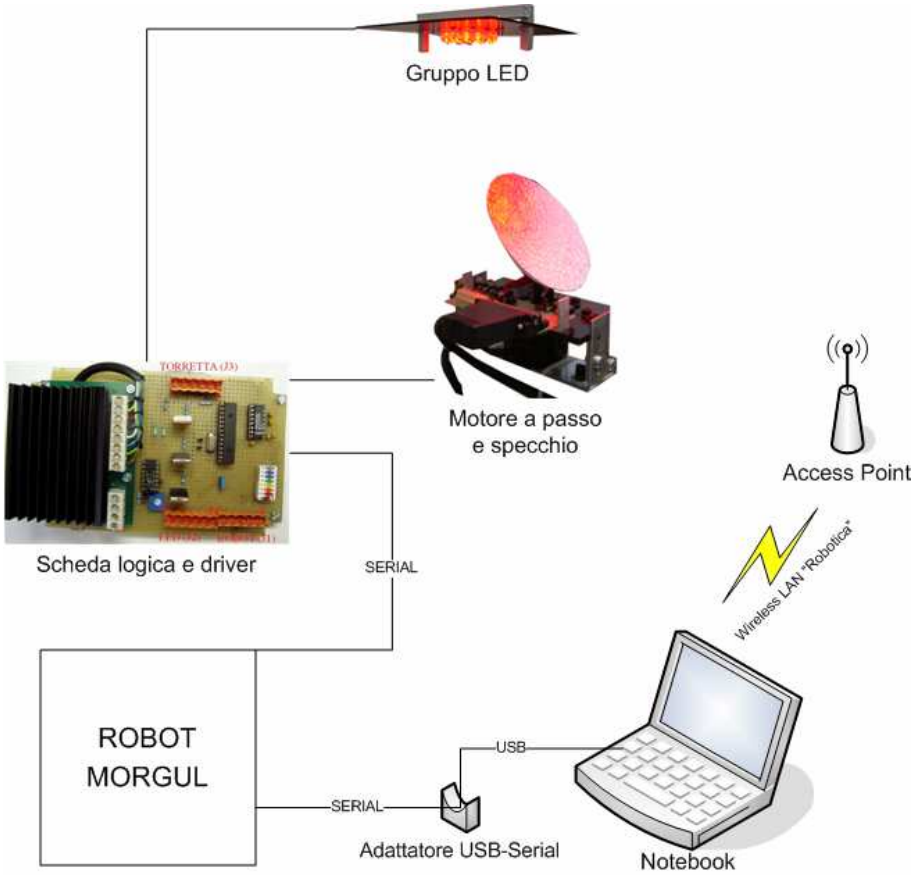

Figura 4 : Schema a blocchi del marker e delle connessioni con Morgul

Lo schema di Figura 4 viene proposto per meglio esemplificare quanto precedentemente descritto: i componenti e le connessioni presenti tra di essi.

Un controllo sui dati impedisce l'inserimento di comandi e valori inesistenti o fuori scala. L'introduzione di un valore errato annulla l'inserimento corrente e richiede un nuovo comando. L'invio di un comando può avvenire in qualsiasi momento, senza che ciò influisca su eventuali operazioni in corso (quindi se sta lampeggiando il gruppo di LEDA posso inviare comandi agli altri gruppi di LED o al motore senza che ciò si rifletta sull'azione corrente). Se viene dato un comando a un apparato che sta già eseguendo un'altra azione l'operazione corrente viene annullata e viene eseguito il nuovo comando. Gli unici momenti in cui la scheda non accetta comandi sono durante l'esecuzione del comando di posizionamento del motore a passo e quando viene attivato l'interruttore presente sulla scheda che attiva i lampeggi con una tempistica predeterminata (33 millisecondi acceso e 42 spento).[5]

3. La soluzione adottata

Abbiamo analizzato le funzioni di libreria per il controllo del marker, implementate principalmente nel file `Marker.cpp` presente sul robot Morgul insieme ai file sorgente di Aria.

Tale libreria consiste nella classe Marker, che va istanziata passando un puntatore ad un oggetto di tipo SerialRobotController (che rappresenta la porta seriale di un robot). Una volta creato l'oggetto di tipo Marker, si possono invocare su di esso metodi che permettono di controllare il marker o di leggerne lo stato. I metodi presenti, in particolare, permettono di accendere e spegnere il motore del marker, di spostarlo ad una determinata posizione, di riportarlo nella posizione iniziale e di attivare o disattivare il lampeggio dei led.

Poiché alcuni comandi possono richiedere alcuni secondi per il completamento (come i comandi di posizionamento), la libreria implementa una logica di controllo non bloccante che permette al programma di proseguire mentre il comando viene completato. Quando viene eseguito il comando successivo, la libreria controlla se nella coda delle risposte ricevute dal PIC é presente il carattere di terminazione del comando precedente: se esso é presente, il nuovo comando può essere eseguito, altrimenti il nuovo comando non viene eseguito.

In realtà é presente anche un meccanismo di timeout che fa in modo che se la libreria non riceve una risposta dal PIC entro un determinato intervallo di tempo, il marker ritorna nello stato READY, così da permettere l'esecuzione dei comandi successivi.

Attraverso un testing accurato della libreria, siamo riusciti a riconoscere alcuni problemi che causavano un funzionamento non corretto del programma in esame.

Prima di tutto, abbiamo notato che una volta che la libreria riceve la risposta di terminazione del comando, si porta in uno stato POST_PAUSE, e attiva un timer con un timeout di 200 ms, scaduto il quale si porta nello stato READY. Probabilmente questo comportamento nasce dall'esigenza di introdurre una piccola pausa tra l'esecuzione di comandi diversi. Tuttavia questo fa in modo che all'esecuzione del secondo comando, la libreria legga la risposta dal PIC, attivi il timer e rifiuti di eseguire il nuovo comando, poiché si trova nello stato POST_PAUSE.

Il problema é stato risolto eliminando lo stato POST_PAUSE alla ricezione della risposta di terminazione; la pausa tra l'esecuzione di due comandi successivi è stata realizzata mediante la funzione bloccante `Aria::sleep()`.

Inoltre é stato introdotto un metodo che facilita il compito di chi scrive programmi utilizzando la libreria del marker; tale funzione (`waitCommandReady()`) permette di attendere finché il comando precedente non é terminato (o é andato in timeout). La funzione é stata realizzata con un semplice ciclo che controlla la terminazione del comando precedente e, in caso negativo, attende 100 ms.

4. Modalità operative

In questo capitolo descriviamo le linee guida per l'utilizzo del software da noi realizzato, presentando una serie di operazioni necessarie per eseguire qualunque software su Morgul, seguite da una dettagliata descrizione dei programmi da noi realizzati per il test delle modifiche di cui abbiamo parlato nel capitolo precedente di questo elaborato.

4.1. Operazioni preliminari

Per il collegamento al robot Morgul usiamo SSH utilizzando l'indirizzo del calcolatore installato sul robot (192.0.2.10). Una volta connessi avendo inserito nome utente e password, il comando da utilizzare per l'accensione del robot è :

```
morgulc -R
```

Usando l'opzione `-r` si spegne il robot, mentre i comandi `-L` e `-l` servono rispettivamente per accendere e spegnere il gruppo di luci presenti sul robot.

In alternativa è possibile usare lo script chiamato `onrobot.sh` (e il suo corrispettivo `offrobot.sh` per lo spegnimento) che esegue alcune operazioni per l'accensione del robot.

Per comandare Morgul nelle situazioni di uscita e ingresso dalla *docking station*, esistono opportuni script chiamati `exitrobot.sh` (che esegue azioni per l'uscita di Morgul dal dock e il suo posizionamento nell'area di lavoro del laboratorio) e `goHome.sh` che esegue operazioni di "parcheggio" di Morgul nella docking station. Va tenuto presente che `goHome.sh` necessita che la parte anteriore del robot sia posizionata rivolta verso la docking station.

Per comandare il robot nell'area di lavoro del laboratorio, infine, esiste lo script `muoveamano.sh` che permette il comando tramite tastiera dei movimenti del robot. Ci siamo avvalsi di tutti gli script qui presentati in modo da posizionare il robot nella zona del laboratorio che ritenevamo più opportuna per testare i programmi di utilizzo del marker.

4.2. Modalità di installazione

Il pacchetto da noi realizzato è composto da due parti: la libreria per il controllo del marker e i programmi di test.

Prima di tutto è necessario decomprimere l'archivio marker.tar.gz con il comando

```
tar xvzf marker.tar.gz
```

Nella cartella `marker` si trovano due sottocartelle: `libMarker` e `test`. Spostandosi nella cartella `libMarker`, è possibile compilare la libreria semplicemente digitando il comando `make`.

Fatto ciò verranno compilati i file `Marker.cpp` e `RobotSerialController.cpp`. Inoltre verrà creato nella sottocartella `lib` la libreria vera e propria (`libMarker.so`).

Digitando il comando `make install` con gli opportuni permessi, la libreria (`libMarker.so`) e gli header C++ (`Marker.h` e `RobotSerialController.h`) verranno installati nello stesso percorso delle librerie ARIA. Inoltre verrà eseguito il comando `ldconfig` che permette al sistema di riconoscere la nuova libreria installata.

Spostandosi nella cartella `test` è quindi possibile compilare i file di test da noi realizzati digitando il comando `make`. A questo punto saranno disponibili i programmi `testmarker1`, `testmarker2` e `serial`, che possono essere eseguiti sul robot. Va ricordato che la connessione seriale tra il calcolatore e il robot è realizzata mediante un adattatore seriale–USB e quindi è necessario passare ai programmi ARIA l'opzione `-rp /dev/ttyUSB0`

Per scrivere un nuovo programma che utilizza il marker, è sufficiente includere nei sorgenti C++ l'header `Marker.h`, istanziare la classe Marker e invocarne i metodi.

Per compilare tale programma è possibile procedere in due modi:

- copiare il Makefile presente nella directory `test`, quindi modificarlo specificando il nome dei file da compilare attraverso la variabile PROGS;
- utilizzare il Makefile standard per un programma basato sulle librerie ARIA; l'unica modifica che si rende necessaria è l'aggiunta dell'opzione `-lMarker` alle opzioni del linker.

4.3. Descrizione dei programmi da noi realizzati per il testing

I file di test scritti durante lo svolgimento di questo progetto sono `testmarker1`, `serial` e `testmarker2` .

Il primo (`testmarker1`) è basato sul programma `testmarker`, già esistente nel momento in cui abbiamo iniziato a lavorare su questo progetto. Il programma esistente avrebbe dovuto eseguire una calibrazione del marker seguita da alcuni comandi di rotazione e accensione delle luci.

In realtà tutto quello che veniva eseguita era l'accensione delle luci: il marker rimaneva poi in una situazione di stallo e non accettava altri comandi né eseguiva altre operazioni.

Il file `testmarker1.cpp`, sorgente del programma di test da noi sviluppato, presenta lo stesso concetto del programma originale, ed è stato usato durante lo svolgimento del lavoro per verificare che le modifiche da noi eseguite alle librerie del marker di Morgul fornissero i risultati desiderati.

Le linee di codice nella prima parte contengono una inizializzazione standard del robot utilizzando le librerie ARIA: creano un nuovo oggetto *ArRobot*, inizializzano la porta seriale, il driver del marker, preparano il robot per la connessione (o segnalano un problema in caso di mancata connessione).

Le successive linee di codice rappresentano una serie di comandi per il test del funzionamento del marker; il programma, in particolare:

- accende la scheda driver del motore a passo mediante il metodo `marker.setMotorPower(Marker::MOTOR_ON);`
- invoca il metodo `marker.motorGoHome()` che porta il motore nella posizione iniziale;
- esegue operazioni di movimento del motore a passo portandolo in posizioni desiderate (per esempio la funzione `marker.setOrientamento(PI / 2.0)` serve a portare il motore ad un angolo di 90 gradi dalla posizione iniziale);
- attiva vari tipi di lampeggio delle luci LED del gruppo A; il metodo `marker.attivaLampeggio(1, 300, 30, 20)` permette di effettuare 300 lampeggi con led accesi per 30 ms e spenti per 20;
- ferma il lampeggio con il metodo `marker.fermaLampeggio()`;
- riporta il motore a passo nella posizione iniziale;
- spegne i led e la scheda driver del motore a passo mediante il metodo `marker.close()`.

Il programma `serial.cpp` rispecchia il precedente nella parte di inizializzazione del robot; il suo funzionamento è un semplice ciclo *while* che legge una stringa di testo da input e la invia al controller del marker restituendo all'utente la risposta della scheda. I comandi che possono essere utilizzati sono documentati nel lavoro di Matteo Manzini. [5]

Il PIC riceve in ingresso dei comandi: possono essere sotto forma di singolo carattere oppure di stringa nel caso di comandi complessi come quelli legati al lampeggio dei LED. Avendo realizzato un software che lavora a basso livello è stato possibile analizzare in dettaglio la risposta del PIC ai comandi da noi inviati; questo ci ha permesso di comprendere meglio il comportamento del PIC in relazione ai comandi da noi inviati.

Il file `testmarker2.cpp`, infine, usato come esempio dimostrativo, esegue una serie di movimenti del motore e diversi tipi di lampeggi con durate e frequenze differenti. Il listato è molto simile a `testmarker1.cpp` e si differenzia nella serie di comandi inviati al marker.

4.4. Avvertenze

In questo capitolo segnaliamo un paio di precisazioni da tenere presente durante l'utilizzo del software realizzato:

- Per un corretto funzionamento dei comandi inviati al PIC, è importante ricordarsi di inserire `marker.waitCommandReady()` per attendere la terminazione del comando in esecuzione sul marker.
- Il comando "p" (che ordina al motore di portare il marker nella posizione iniziale) dato alla scheda driver del marker non restituisce nulla se il motore a passo si trova già nella posizione "home". Abbiamo quindi abbassato il tempo di timeout, scaduto il quale il programma torna

nello stato READY e accetta l'invio di nuovi comandi. Non ricevendo risposta dal PIC consideriamo quindi il motore a passo già nella posizione iniziale.

- Esiste, ad oggi, un solo gruppo di LED (il gruppo A) anche se nel PIC è predisposto supporto per 3 diversi gruppi di led. Per questo motivo i comandi inviati ai gruppi B e C non vanno utilizzati.

5. Conclusioni e sviluppi futuri

Al termine di questo lavoro possiamo dire che le modifiche da noi apportate alle librerie mettono a disposizione funzioni stabili e documentate per la gestione del marker del robot Morgul. Possiamo dire di avere portato a termine quanto richiesto all'inizio di questo progetto, cioè la rimessa in funzione, la prova e documentazione del marker sul robot Morgul.

Tra gli sviluppi futuri che ci sentiamo di proporre c'è la realizzazione di un'interfaccia grafica basata sulle librerie ARIA che sfrutti il lavoro da noi svolto.

Bibliografia

[1] Cassinis, R., Tampalini, F.: "AMIRoLoS - An Active Marker Internet-based Robot Localization System", Robotics and Autonomous Systems, Vol. 55, N. 4, Pag. 306-315, Elsevier, Amsterdam, The Netherlands, 2007

[2] http://www.mobilerobots.com/ , sito ufficiale di ActivMedia, designer e realizzatori dei robot Pioneer

[3] "Pioneer 2 & Pioneer 3 Operating Manual2" , Activmedia

[4] Gentile R., Girelli B. B., Zanuzzi M., "Marker", 26 settembre 2005.

[5] Manzini, M. : "Sviluppo di un supporto hardware e software per la gestione di un marker e di un motore a passo", 30 maggio 2005.

Indice