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)

Programma per la creazione di mappe con il sensore UHG-08LX su Morgul

Elaborato di esame di:

Davide Simoncelli, Nicolò Facchi, Nicola Mazzini e Michele Scandale

Consegnato il: **1 luglio 2011**

Sommario

L'obiettivo del lavoro è sviluppare un software che permetta di esplorare l'ambiente e ricavarne una mappa mediante l'utilizzo del sensore UHG-08LX di Hokuyo. L'ambiente è esplorato dal robot cercando di seguire un percorso specificato dall'utente. La mappa infine ottenuta sarà visualizzabile e modificabile con gli strumenti del framework Aria, come Mapper3Basic.

1. Introduzione

In questa relazione descriviamo il lavoro svolto per il corso di Robotica Mobile.

Il robot utilizzato è Morgul, un robot Pioneer della serie 3: è in grado di raggiungere una velocità massima di 700 mm/s ed è dotato di 16 sonar, 8 davanti e 8 dietro [5]. Per questo lavoro il robot è dotato anche del sensore laser UHG-08LX della casa giapponese Hokuyo.

Lo scopo del lavoro è far seguire al robot un percorso e scansionare l'ambiente in punti predefiniti. L'ambiente di test utilizzato è stato il laboratorio di Robotica Avanzata della Facoltà di Ingegneria.

2. Il problema affrontato

Il problema prevede la descrizione da parte dell'utente di un percorso da far seguire al robot e dei punti nei quali è necessario scansionare l'ambiente con il sensore laser.

Il robot deve cercare di seguire il più fedelmente possibile il percorso specificato evitando eventuali ostacoli presenti sul percorso. Ovviamente le precedenti richieste sono in contrapposizione tra loro ed è quindi necessario stabilire un trade-off fra le due esigenze contrapposte.

I dati raccolti dal sensore laser devono essere poi elaborati per produrre una mappa con risoluzione di 1 mm dell'ambiente scansionato che sia utilizzabile dagli strumenti software messi a diposizione da Aria. Volendo ottenere una risoluzione di 1 mm ed essendo necessario elaborare una grossa mole di dati derivanti da più scansioni, è necessario avere a diposizione risorse di calcolo e memoria sufficienti ad affrontare il problema. Infatti dopo una prima analisi, è risultato che i dati elaborati di una singola scansione occupano in media 30 MB.

3. La soluzione adottata

Per risolvere il problema delle risorse computazionali, è stato deciso di spostare la fase di elaborazione su un calcolatore diverso da quello presente su Morgul; per questo motivo il software sviluppato presenta due componenti: una parte client e una parte server.

3.1. Il componente client

Le responsabilità di questo componente sono:

- far seguire al robot il percorso specificato;
- gestire la scansione e il salvataggio dei dati scansionati;
- inviare i dati al componente server[1].

3.1.1. La specifica del percorso

Per la prima responsabilità è necessario innanzitutto specificare il percorso da fare seguire al robot; per questo motivo è stato adottato un file di configurazione che contiene una serie di istruzioni, una per ogni riga. Il sottocomponente che si occupa di analizzare il file del percorso richiede che siano soddisfatte le seguenti precondizioni:

- il file deve contenere un'istruzione per riga;
- sono consentite righe vuote, che saranno ignorate;
- il file di configurazione può contenere solo tre tipi di istruzioni:
  - move <valore>: permette di far traslare in avanti il robot del valore specificato in millimetri; il valore deve essere maggiore o uguale a 5 mm;
  - dh <valore>: permette di fare un movimento di rotazione. Il valore specifica l'angolo in gradi e deve essere compreso fra -180° e 180°; valori positivi implicano una rotazione in senso antiorario e quelli negativi in senso orario;
  - scan: fa eseguire al robot una scansione mediante il sensore laser.
- non sono ammessi spazi vuoti all'inizio e alla fine della riga e fra il nome istruzione e il valore deve esserci solo uno spazio.

Per esempio si vuole far compiere al robot uno spostamento in avanti di un metro, una scansione, una rotazione in senso antiorario di 90°, uno spostamento in avanti di mezzo metro e infine un'altra scansione. Il percorso risultante è il seguente:

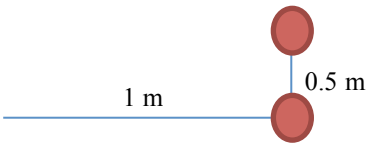


Figura 1 – Esempio di percorso

Il file di configurazione per fare compiere il movimento è il seguente:

move 1000

scan

dh 90

[1] Questa scelta di inviare i dati da client a server è derivata dall'esigenza di semplificare l'implementazione. Questo comportamento è paragonabile all'utilizzo del protocollo FTP quando il client ha l'esigenza di mandare dei dati al server, con la differenza che nel nostro caso la porta è unica e fissata.

move 500

scan

Il secondo problema è fare eseguire al robot il percorso descritto nel file; per questo motivo è stato deciso di utilizzare tre gruppi di azioni, poiché il comportamento dipende dall'istruzione corrente che si sta eseguendo. Dato che ogni gruppo implementa il comportamento di un tipo di istruzione, i gruppi sono mutamente esclusivi. Inoltre tale decisione permette di risolvere il trade off fra far percorrere al robot un percorso dato e, nello stesso tempo, evitare ostacoli. Di seguito sono descritti i tre gruppi di azione individuati.

3.1.1.1 Move action group

Questo gruppo ha tre azioni: l'azione *MovePathAction* che fa compiere al robot un movimento di traslazione pari al valore specificato nel file di configurazione per l'istruzione corrente e due azioni istanza della classe *ArActionAvoidFront* che fanno evitare al robot eventuali ostacoli presenti sul percorso.

Nel caso in cui non siano presenti ostacoli, l'unica azione che interviene è *MovePathAction* che ha la responsabilità di regolare la velocità di traslazione in funzione della distanza percorsa e della distanza ancora da percorrere; se la distanza da percorrere è inferiore a 500 mm, la velocità massima è limitata a 50 mm/s. Di seguito è illustrato il grafico della velocità nel caso in cui si percorra una distanza superiore a 500 mm:

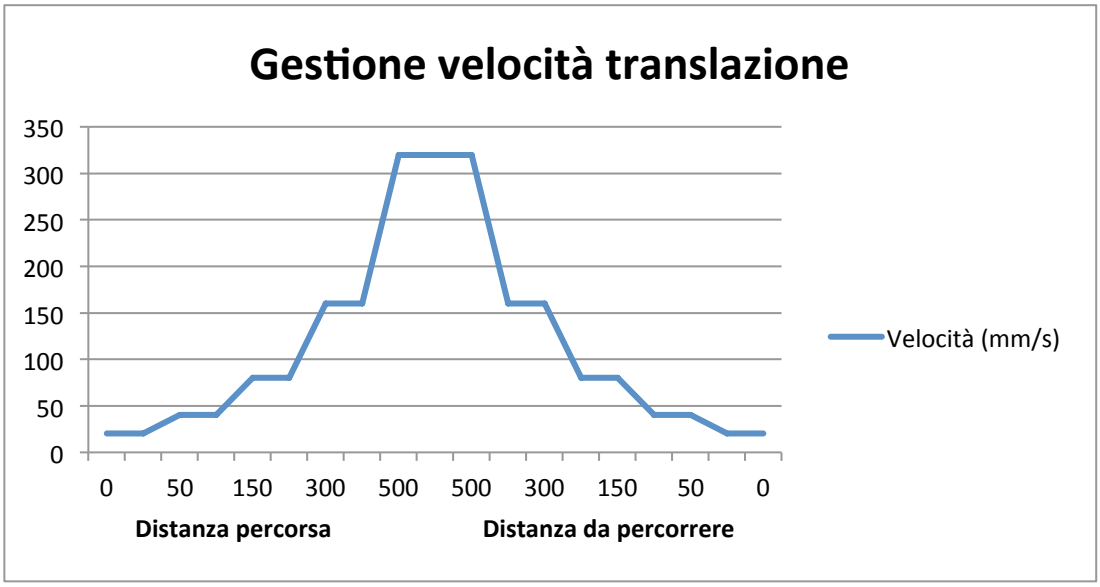


Figura 2 – Grafico della velocità assunta dal robot durante una traslazione

Invece nel caso in cui siano presenti ostacoli, non è possibile prevedere come questi siano disposti sul percorso e quindi il movimento è gestito dalle due azioni istanza della classe *ArActionAvoidFront*. Sono state utilizzate due azioni istanza della classe *ArActionAvoidFront* per differenziare il comportamento del robot nel gestire ostacoli a diverse distanze. Un'azione gestisce gli ostacoli "lontani", cioè quelli che si trovano a 250mm, riducendo la velocità a 100 mm/s ed effettuando una rotazione di 10°. L'altra azione gestisce gli ostacoli "vicini", cioè quelli che si trovano a 100mm, azzerando la velocità di traslazione del robot ed effettuando una rotazione di 45°. Il movimento è considerato terminato quando il robot compie una traslazione pari al valore specificato dall'istruzione di movimento; in questo caso il movimento non sarà rettilineo.

3.1.1.2 Rotate action group

Questo gruppo ha una sola azione chiamata *RotatePathAction*, che permette di compiere un movimento di rotazione gestendo la presenza di ostacoli ai lati del robot. Nel caso siano presenti ostacoli che impediscono al robot di compiere tutto l'angolo di rotazione specificato, il movimento è considerato terminato e si passa all'istruzione successiva. L'azione gestisce anche la velocità di rotazione: se la rotazione da percorrere è inferiore a 25°, la velocità è impostata a 5°/s, altrimenti si regola la velocità come indicato dal seguente grafico:

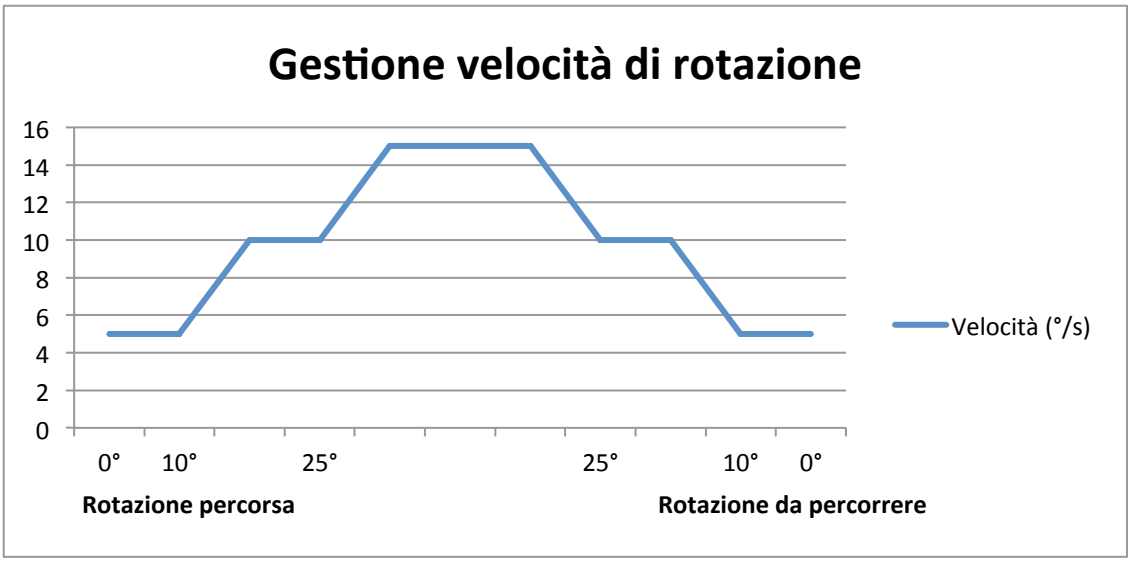


Figura 3 - Grafico della velocità assunta dal robot durante una rotazione

3.1.1.3 Scan action group

Questo gruppo contiene l'azione *ScanEnviromentAction* che effettua la scansione dell'ambiente e converte le coordinate x e y della posa del centro del robot nelle coordinate x e y del sensore laser. Infatti quando il robot ruota, il laser compie una rototraslazione.

Il sensore laser è spostato rispetto alla coordinata x del centro del robot di 23,5 cm come mostrato dalla figura seguente:

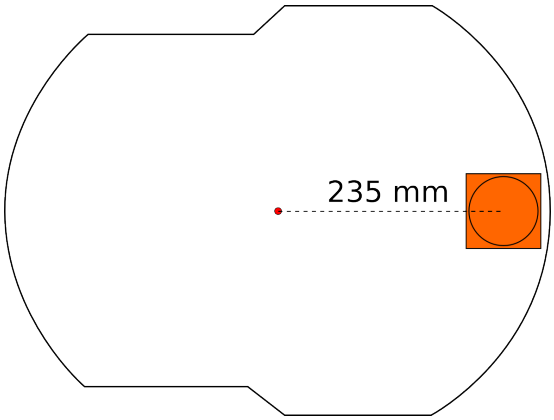


Figura 4 – Posizione del laser rispetto al centro del robot

Una volta concluso un movimento e prima di effettuare ogni scansione, si attendono circa 2,5 secondi in modo tale che la struttura non sia soggetta a vibrazioni e quindi evitare che le misure siano affette da ulteriori errori.

Per ogni scansione è creato e salvato sul calcolatore di Morgul un file contenente i dati della scansione nel seguente formato: la prima riga specifica la distanza massima rilevabile dal sensore, le tre righe successive contengono rispettivamente la coordinata x, y e l'angolo di rotazione del robot e le righe seguenti contengono i valori delle scansioni; ogni valore è costituito dalla distanza rilevata e dall'angolo da cui è stata rilevata tale distanza separati da uno spazio.

3.1.2. L'invio dei dati

Una volta che il percorso è stato completato, il client può inviare i dati al server. Per ogni file viene generato un pacchetto nel seguente formato:

max_distance	x	y	angle	data_length	distance	angle	…

- max_distance (4 byte): è la massima distanza rilevabile dal sensore laser;
- x (4 byte): è la coordinata x del sensore;
- y (4 byte): è la coordinata y del sensore;
- angle (4 byte): è l'angolo di rotazione del sensore;
- data_length (4 byte): è il numero di letture fatte dal laser che permette di determinare la lunghezza del campo dati del pacchetto;
- distance (4 byte) e angle (4 byte): ogni coppia (distance, angle) rappresenta una lettura del sensore.

Nel caso in cui non sia possibile contattare il server, poiché per esempio il robot non risulta essere temporaneamente collegato alla rete, i file delle letture vengono mantenuti in locale ed è possibile inviarli successivamente al server.

3.1.3. La struttura

Di seguito è presentato il diagramma delle classi del componente client:

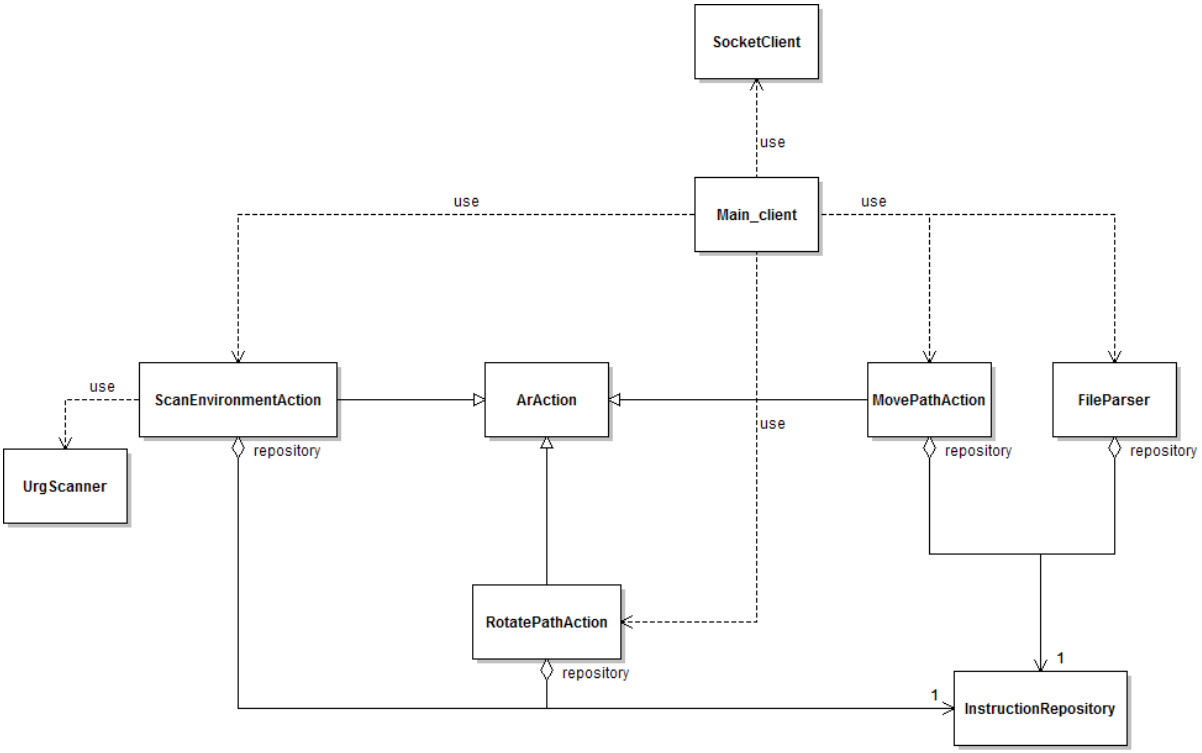


Figura 5 – Diagramma delle classi componente client

- SocketClient: la classe mette a diposizione i metodi per gestire la comunicazione con il server;
- UrgScanner: la classe permette di interagire con il sensore laser;
- FileParser: la classe fa l'analisi del file di configurazione che contiene la specifica del percorso;
- InstructionRepository: gestisce la coda delle istruzioni e descrive la macchina a stati per escludere il comportamento dei gruppi di azioni;
- RotatePathAction: la classe gestisce il comportamento per l'istruzione di rotazione;
- MovePathAction: la classe gestisce il comportamento per l'istruzione di movimento rettilineo;
- ScanEnviromentAction: la classe gestisce la scansione dell'ambiente.

Per approfondimenti sulla struttura, riferirsi alla documentazione del codice contenuta nella cartella doc/client del progetto.

3.2. Il componente server

Le responsabilità di questo componente sono:

- ricevere i dati dal componente client e gestirli in apposite strutture;
- creare la mappa-database;
- creare la mappa nel formato accettato dai software delle librerie Aria.

3.2.1. Mappa-database

La mappa-database è una struttura dati che svolge due compiti fondamentali:

- contenitore per i dati in ricezione;
- unità di elaborazione dei dati stessi mediante un database sqlite3 temporaneo.

3.2.1.1 Ricezione dei dati

Per la fase di ricezione, i dati sono letti dal socket, che rappresenta la connessione tra il client e il server. Ogni misura letta viene opportunamente trasformata dal formato di rete al formato locale e viene immagazzinata in una lista di misure che alla fine della comunicazione saranno elaborate.

3.2.1.2 Elaborazione

Questa seconda fase inizia alla chiusura della comunicazione tra client e server.

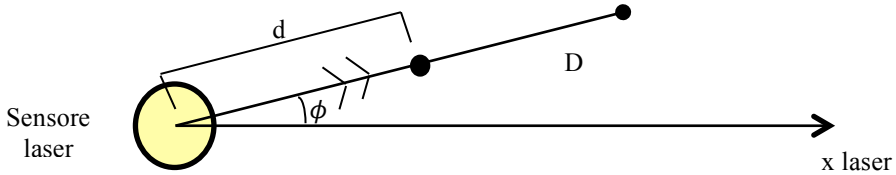
Viene creato un database temporaneo sqlite3 su file, con un'unica tabella organizzata con le seguenti colonne:

1. Id: identificatore della scansione;
2. X: coordinata x di un generico punto dell'ambiente interessato, derivato da un opportuno calcolo fatto con le misure delle scansioni.
3. Y: coordinata y di un generico punto dell'ambiente interessato, derivato da un opportuno calcolo fatto con le misure delle scansioni;
4. Value: rappresenta lo spazio libero (valore 0) oppure uno spazio occupato (valore 2).[2]

Per ogni lettura si ha a disposizione la distanza *D* rilevata dal sensore e l'angolo ϕ a cui tale distanza è stata misurata. I valori inseriti nella tabella vengono ricavati nel seguente modo:

- si considera un segmento che ha un estremo nel centro del sensore, lunghezza di valore *D* e direzione data dall'angolo ϕ;
- si ricavano le coordinate *x* e *y* di ogni punto del segmento con risoluzione al millimetro, iterando con un indice *d* compreso fra 1 e *D* e applicando le seguenti due formule:
  - $x = x_0 + d \cdot \cos(\phi + \vartheta_0)$
  - $y = y_0 + d \cdot \sin(\phi + \vartheta_0)$

  dove il punto (x_0, y_0, ϑ_0) identifica la posizione del laser rispetto al sistema di riferimento, che ha come origine la posizione iniziale del centro del robot. La mappa infatti verrà costruita rispetto a tale sistema di riferimento;
- per ogni punto (*x, y*) ricavato, si assegnano i valori al campo Value nel seguente modo: per i valori di distanza $1 \leq d < D$ si assume che i punti interessati siano spazio libero e quindi si assegna il valore 0, mentre per i punti a distanza *D*, con $D < distanza\_massima$ (11 m per il sensore utilizzato), si considera lo spazio occupato e si assegna il valore 2.



La scelta di utilizzare un database semplifica la costruzione della mappa globale a partire da quelle parziali, grazie all'utilizzo delle funzioni aggregate di SQL e inoltre consente di gestire un numero di scansioni a piacere, permettendo di analizzare uno spazio la cui dimensione è a priori sconosciuta.

[2] La scelta dei valori 0 e 2 deriva dalla convenzione usata in un precedente lavoro di Mazzini nel quale la scelta era stata dettata dalla necessità di gestire anche un terzo valore rappresentante lo stato di incertezza e quindi realizzato con l'intero 1.

3.2.2. Mappa per i software della libreria Aria

Per generare una mappa, vengono inseriti tutti quei punti che sono valutati come ostacoli. Dato che una stessa coordinata (x, y) può comparire più volte nella tabella, vengono considerati ostacoli i punti per cui la media pesata dei valori associati è superiore a 1,5. Questo corrisponde a chiedere che almeno il 75% dei valori relativi a una coppia di coordinate abbia valore pari a 2.

La mappa generata è una nube di punti, in cui è mantenuta una risoluzione al millimetro e questo rende accettabile l'utilizzo di una mappa così fatta; inoltre delle icone evidenziano la posa in cui sono state effettuate le scansioni.

3.3. Mappa di esempio

Di seguito è presentato un esempio nel quale si vuole creare una mappa del laboratorio di Robotica Avanzata. L'insieme di istruzioni che descrivono il percorso fatto percorrere al robot è il seguente:

scan

dh 180

scan

move 1900

scan

dh 90

scan

move 1200

scan

dh -90

scan

dh 90

move 1900

scan

dh 90

scan

move 2000

scan

move 2000

scan

move 1500

scan

dh 90

scan

dh 180

scan

Il robot eseguirà 13 scansioni dell'ambiente e il percorso eseguito è rappresentato dalla seguente figura, in rosso sono rappresentati i punti in cui il robot è fermo e ogni freccia indica una scansione con la relativa direzione. Più frecce uscenti da uno stesso punto indicano scansioni intervallate dal solo movimento di rotazione:

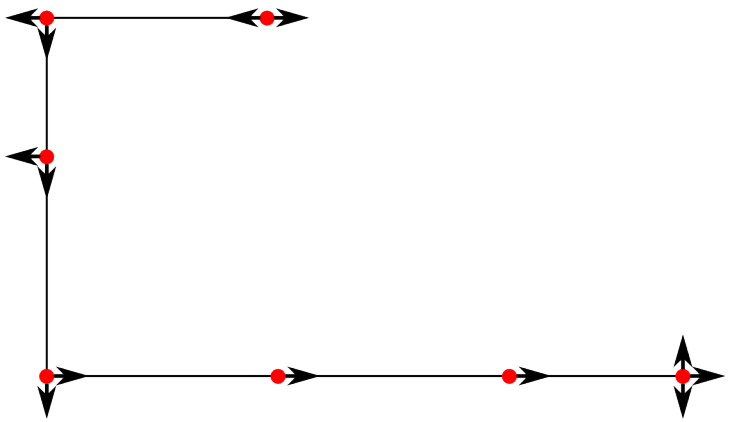


Figura 6 – Percorso eseguito dal robot e relative scansioni

Infine la mappa ottenuta a seguito dell'elaborazione dei dati delle scansioni è la seguente:

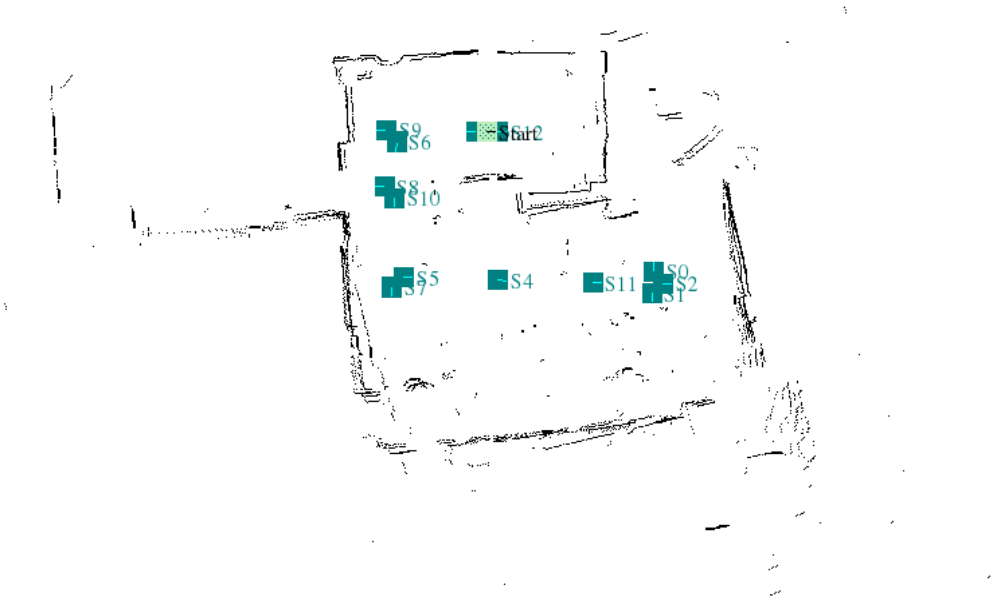


Figura 7 – Mappa ottenuta a seguito delle elaborazione delle scansioni

La numerazione delle etichette che indicano le pose del robot al momento della scansione è influenzata dall'ordine in cui vengono inviati i dati delle scansioni dal client al server. Questo potrebbe non coincidere con l'ordine con cui sono state effettivamente eseguite le scansioni, come è possibile notare nell'esempio sopra proposto.

4. Modalità operative

Il sistema software sviluppato per la realizzazione di mappe è costituito da due componenti indipendenti, una la parte client e l'altra la parte server.

Il client viene eseguito sul calcolatore del robot Morgul e il server invece è consigliabile eseguirlo su un calcolatore differente poiché le risorse computazionali necessarie per l'elaborazione dei dati del sensore non sono disponibili sul calcolatore del robot Morgul.

I file di progetto sono organizzati nella seguente struttura:

MorgulMapping\

client: contiene i sorgenti per il client e il Makefile per compilarlo

server: contiene i sorgenti per il server e il Makefile per compilarlo

doc: contiene la documentazione del sistema software generata con Doxygen

client: contiene la documentazione del componente client

server: contiene la documentazione del componente server

include: contiene gli header file condivisi con i componenti client e server

Makefile: Makefile per compilare i componenti client e server

Durante il lavoro svolto sono stati utilizzati i seguenti calcolatori: Acer sul robot Morgul (Processore Intel Pentium4 1,6 GHz e memoria 256MB) e un nostro calcolatore (Intel i7 3,4 GHz e memoria 4 GB).

Di seguito supponiamo di partire da una distribuzione Debian a 32 bit appena installata sulla quale siano installati i seguenti programmi: compilatore gcc, g++, i comandi make, wget e unzip. Nel caso in cui non siano installati procedere con il seguente comando:

```
sudo apt-get install wget unzip make g++ gcc
```

4.1. Componenti necessari

4.1.1. Componenti lato client

Il componente hardware utilizzato è il sensore UHG-08LX della casa giapponese Hokuyo con le seguenti caratteristiche:

- **distanza di rilevamento**: 30 mm - 11000 mm
- **accuratezza**: da 30 mm a 1000 mm ± 30 mm, da 1000 mm a 8000 mm ± 3%
- **angolo di rilevamento**: 270°
- **risoluzione** 0.36°
- **tempo di scansione**: 67ms per scansione
- **interfaccia**: USB 2.0
- **protocollo di comunicazione**: SCIP 2.0
- **alimentazione**: 12 V DC

Per far funzionare il sensore è necessario installare il driver CDC ACM, che permette la connessione con il laser scanner e le librerie urg versione 0.8.12, che permettono di interagire con il dispositivo laser. Inoltre è necessario installare le librerie Boost e SDL, che sono delle dipendenze per potere installare le librerie urg [1].

Infine sono richieste le librerie Aria, che permettono la gestione del robot Morgul.

4.1.2. Componenti lato server

È necessario installare le librerie di SQLite per memorizzare i dati relativi alla mappa e opzionalmente le librerie Aria insieme a Mapper3Basic e MobileSim per sfruttare successivamente la mappa generata.

4.2. Modalità di installazione

4.2.1. Installazione lato client

Per installare i driver CDC ACM per il corretto funzionamento del sensore laser, è necessario ricompilare il kernel del sistema abilitando l'opzione *USB Modem (CDC ACM Support)* nel file di configurazione del kernel. Per fare questo si procede nel seguente modo:

scaricare i sorgenti del kernel per la versione attualmente in uso sul sistema con il comando:

```
sudo apt-get install linux-headers-$(uname –r)
```

Installare le librerie grafiche ncurses-devel che servono per il menu grafico per accedere alla configurazione del kernel:

```
sudo apt-get install libncurses5-dev
```

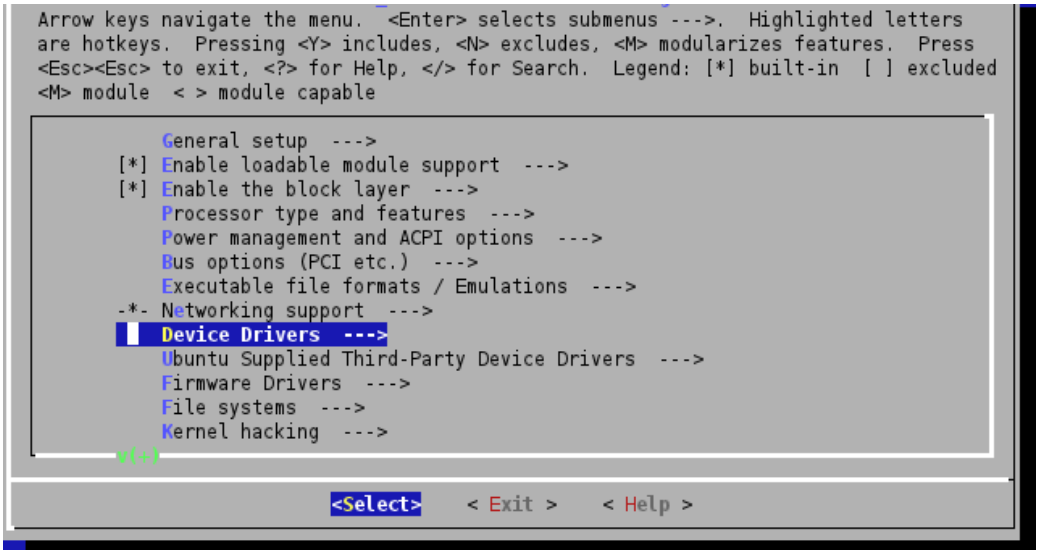
Procedere a compilare il kernel; è necessario andare nella cartella contenente i sorgenti del kernel scaricati in precedenza con il comando:

```
cd /usr/src/linux-headers-$(uname -r)
```

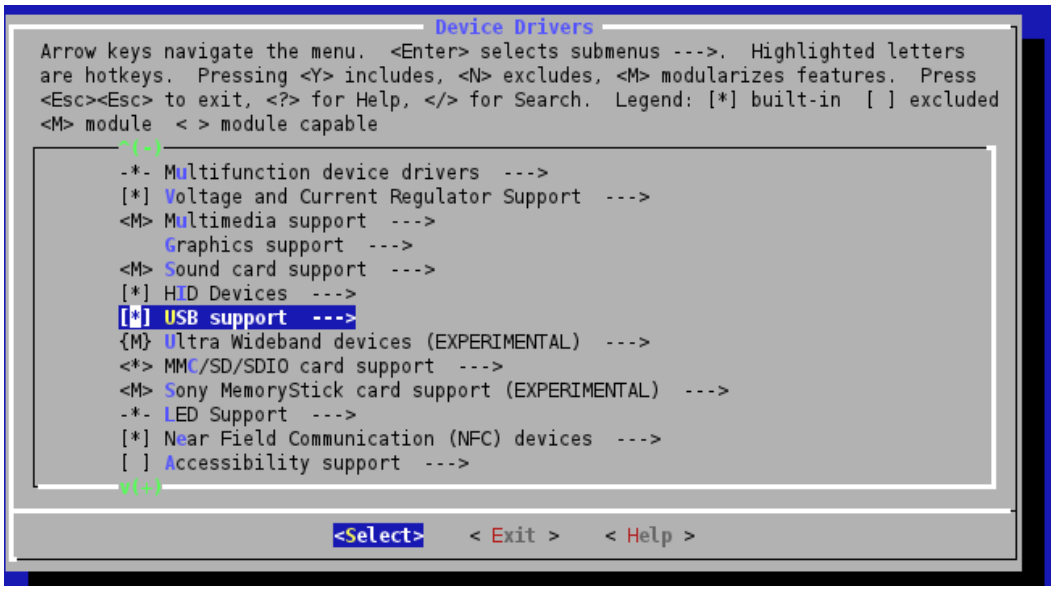
Infine invocare il menu grafico di configurazione con il comando:

```
sudo make menuconfig
```

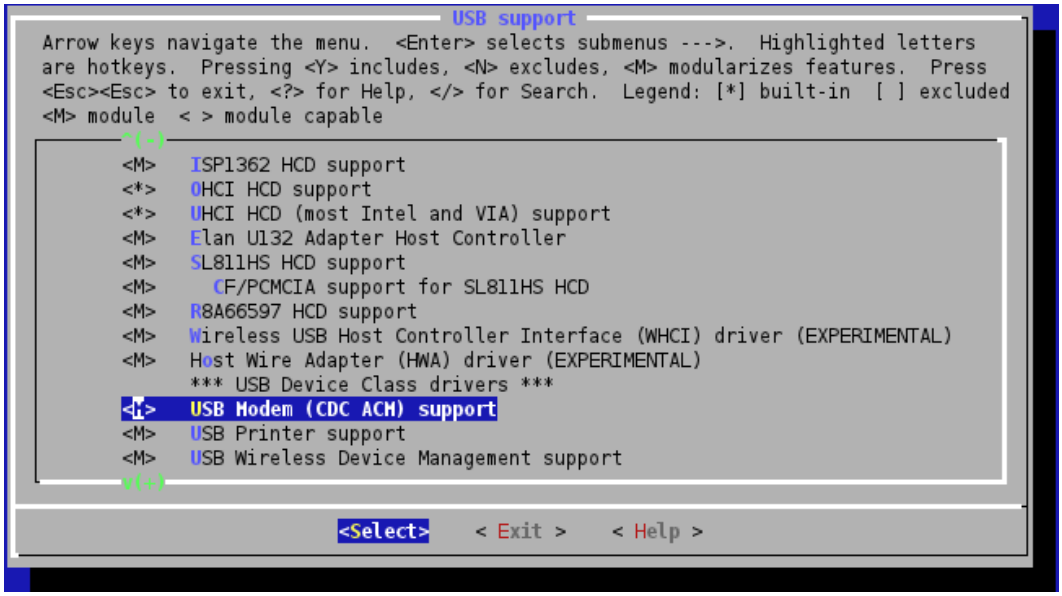
Una volta visualizzato il menu grafico, si selezioni la voce *Device Drivers*:



In seguito la voce USB *Support*:



Infine si selezioni la voce USB Model (CDC ACM Support) e si clicchi il tasto M per compilare il driver come modulo:



Ora si proceda con la compilazione e l'installazione del modulo:

```
sudo make modules_install
```

e si carichi il modulo appena installato:

```
sudo modprobe cdc-acm
```

Successivamente si proceda con l'installazione delle librerie urg che però richiedono l'installazione delle dipendenze Boost e SDL con il comando:

```
sudo apt-get install libsdl1.2-dev libsdl-net1.2-dev libboost-dev libboost-regex-dev
```

Ora ci si porti nella home directory (cd ~) e si proceda allo scaricamento dei sorgenti della libreria urg con il comando:

```
wget http://www.hokuyo-aut.jp/02sensor/07scanner/download/urg_programs_en/urg-0.8.12.zip
```

Si estragga il contenuto dell'archivio ZIP appena scaricato e ci si porti nella directory appena creata:

```
unzip urg-0.8.12.zip
cd urg-0.8.12/
```

e si proceda con l'installazione:

```
./configure
make
sudo make install
```

Ora si ritorni alla home directory e si proceda con lo scaricamento delle librerie Aria:

```
wget http://robots.mobilerobots.com/ARIA/download/current/libaria_2.7.2_i386.deb
```

e si proceda all'installazione con il commando *dpkg*:

```
sudo dpkg -i libaria_2.7.2_i386.deb
```

Una volta terminata l'installazione delle dipendenze, è possibile procedere alla compilazione del client; supponendo di trovarci nella cartella *MorgulMapping* ci si porti nella cartella client e si esegua il comando *make*. Questo genererà l'eseguibile *client_map*.

4.2.2. Installazione lato server

Per installare le librerie SQLite è necessario eseguire il seguente comando:

```
sudo apt-get install libsqlite3-dev
```

In seguito ci si porti nella home directory (cd ~) e si proceda quindi ad installare Aria:

```
wget http://robots.mobilerobots.com/ARIA/download/current/libaria_2.7.2_i386.deb

sudo dpkg -i libaria_2.7.2_i386.deb
```

Ed infine si installino MobileSim e Mapper3Basic:

```
wget
http://robots.mobilerobots.com/MobileSim/download/current/mobilesim_0.5.0_i386.de
b

  sudo dpkg -i mobilesim_0.5.0_i386.deb

wget http://robots.mobilerobots.com/Mapper3Basic/download/current/mapper3-
basic_2.2.5-1_i386.deb

sudo dpkg -i mapper3-basic_2.2.5-1_i386.deb
```

Una volta terminata l'installazione delle dipendenze, è possibile procedere alla compilazione del server; supponendo di trovarci nella cartella *MorgulMapping* ci si porti nella cartella server e si esegua il comando *make*. Questo genererà l'eseguibile *server_map*.

4.3. Modalità di esecuzione

4.3.1. Esecuzione lato client

La sintassi per il comando client_map è la seguente:

```
client_map -dir path_to_dir [-path path_to_path_file [-port sensor_port]] [-sa
server_address -sp server_port]
```

Questo software deve essere eseguito sul robot che si assume sia equipaggiato con laser e sonar. Utilizzare i parametri senza i relativi attributi può portare a malfunzionamenti. Il software presenta tre modalità di funzionamento:

- **esecuzione percorso e scansioni**: Se si desidera che il robot esegua il percorso e le scansioni definite e che i file contenenti i dati delle scansioni vengano salvati nella directory specificata,

ma senza che quest'ultimi vengano inviati al server è necessario utilizzare solo i parametri *-dir* e *–path;*

- **invio dati al server***:* Se si desidera che il client invii al server i file contenuti nella directory specificata, ma senza che il robot esegua nessuna operazione (ad esempio perché c'è la necessità di recuperare dei file contenenti i dati delle scansioni di un esecuzione precedente del programma che non erano ancora stati inviati al server) allora è necessario utilizzare i parametri *-dir*, *-sa* e *-sp*, ma non bisogna utilizzare il parametro *–path;*
- **esecuzione percorso e scansioni e invio dati al server**: se si desidera che il robot esegua il percorso e le scansioni definite e che i file contenenti i dati delle scansioni vengano inviati al server è necessario utilizzare i parametri *-path*, *-dir*, *-sa* e *-sp*. In quest'ultimo caso, se dovessero sorgere problemi nella comunicazione con il server, i file saranno comunque mantenuti in locale sul client in modo di poterli recuperare successivamente.

I parametri da utilizzare sono i seguenti:

- -dir path_to_dir: specifica la directory nella quale si trovano i file contenenti i dati letti dal sensore. Il parametro è obbligatorio.

➢ **Se la directory non esiste verrà creata. In ogni caso é necessario possedere i permessi necessari per poter scrivere e leggere nella directory. Inoltre tutti i file presenti nella directory verranno inviati al server ed eliminati una volta conclusa con successo la trasmissione.**

- -path path_to_path_file: percorso al file di configurazione che specifica le operazioni che devono essere eseguite dal robot.
- -port sensor_port: porta su cui é connesso il sensore laser. Di default é utilizzata /dev/ttyACM0.
- -sa server_address: indirizzo IP del server a cui vanno inviati i dati. Se si utilizza questo parametro é necessario utilizzare anche -sp.
- -sp server_port: porta sulla quale è in ascolto il server. Se si utilizza questo parametro é necessario utilizzare anche -sa.

Di seguito verranno brevemente illustrati tre esempi per il funzionamento del client.

Nel primo si suppone di fare eseguire un percorso al robot e di scansionare l'ambiente tramite un file di configurazione denominato Azioni.txt; i dati per costruire la mappa non vengono inviati al server, ma saranno mantenuti in locale. La directory da utilizzare é *Directory1* (relativa alla posizione dove viene eseguito il software). Quindi il comando da eseguire è il seguente:

```
client_map -dir Directory1 -path Azioni.txt
```

Il secondo esempio invece, permette di inviare i dati di una precedente scansione al server delle mappe. I dati sono contenuti nei file all'interno della directory *Directory1* (in questo caso specificata con un path assoluto) e si suppone che il server sia in ascolto sull'indirizzo IP 132.145.1.3 e sulla porta 6666. In questo caso non passando il parametro -path il robot non esegue nessuna azione.

Questo può essere utile nel caso si vogliano recuperare dei dati che erano stati raccolti in precedenza, ma non ancora inviati al server. Il comando da eseguire è il seguente:

```
client_map -dir /home/utente/Documenti/Directory1 -sa 132.145.1.3 -sp 6666
```

Nel terzo e ultimo esempio si suppone di inviare direttamente i dati delle scansione ad un server in ascolto sull'indirizzo IP 132.145.1.3 e sulla porta 6666. È necessario specificare la directory dove verranno salvati i file contenenti i dati delle scansioni che saranno eliminati una volta inviati con successo al server. Il comando da eseguire è il seguente:

```
client_map -dir Directory1 -path ./Azioni.txt -sa 132.145.1.3 -sp 6666
```

4.3.2. Esecuzione lato server

La sintassi per il comando server_map è la seguente:

```
server_map -sp server_port [program_name]
```

Questo software deve essere eseguito sul calcolatore che esegue le elaborazioni delle scansioni.

I parametri da utilizzare sono i seguenti:

- -sp server_port: porta sulla quale è in ascolto il server;
- -program_name: path completo al programma che deve caricare la mappa.

Per esempio per mettere in ascolto il server sulla porta 6666 e caricare la mappa generata con Mapper3basic, si esegua il seguente comando:

```
server_map –sp 6666 /usr/local/Mapper3Basic/bin/ Mapper3Basic
```

4.4. Avvertenze

Una prima avvertenza è relativa ad un problema con le librerie urg nel linguaggio C++: la funzione di disconnessione non spegne il sensore. Questo problema però non è presente con le librerie in C.

Una seconda nota è invece relativa alla costruzione della mappa, perché sia il laser sia il robot commettono errori di misura: il primo ha un errore potenziale massimo del 3% per ogni metro misurato, e il secondo dipende dall'odometria e dal percorso compiuto. La loro combinazione si ripercuote quindi sulla bontà della mappa finale.

5. Conclusioni e sviluppi futuri

Il presente elaborato ha raggiunto tutti gli obiettivi prefissati, poiché attraverso la navigazione del robot in un ambiente e l'integrazione del laser è stato possibile realizzare una mappa, espressa come nube di punti, dell'area interessata.

Possibili sviluppi futuri del lavoro consistono nel perfezionamento della generazione della mappa mediante l'utilizzo di linee, deducendole dai punti identificati come spazio occupato; inoltre si può proseguire nella direzione che porta alla creazione di un sistema SLAM, in cui il robot è individuato all'interno dell'ambiente mentre si sta muovendo nello stesso.

Bibliografia

[1] Mazzini, N.: “Progetto e realizzazione di un sistema software per la costruzione di una mappa di ambienti mediante laser range scanner”, febbraio 2010.

[2] Mobile Robots: “Documentazione libreria Aria”, http://www.ing.unibs.it/~arl/.

[3] Hokuyo Automatics: “Documentazione librerie per sensore URG”, http://www.hokuyo-aut.jp/02sensor/07scanner/download/urg_programs_en/index.html.

[4] SQLite: “SQLite documents”, http://www.sqlite.org/docs.html

[5] MobileRobots: “Pioneer 3 DX/AT Research Robot Bases”, http://www.mobilerobots.com/PDFs/P3ATDX%20Datasheet.pdf

Indice

Indice delle figure