UNIVERSITÀ DI BRESCIA
FACOLTÀ DI INGEGNERIA
Dipartimento di Ingegneria dell'Informazione

Laboratorio di Robotica Avanzata
Advanced Robotics Laboratory

Corso di Robotica Mobile
(Prof. Riccardo Cassinis)

Gestione di robot multipli tramite central server

Elaborato di esame di:

Massimo Bono, Davide Bonusi, Dario Pellegrini

Consegnato il: **10 giugno 2014**

Sommario

Il lavoro svolto ha come obiettivo quello di utilizzare due o più robot in contemporanea da un unico calcolatore. Per fare ciò i robot fanno riferimento a un server centrale. Per la loro visualizzazione e manipolazione è stato utilizzato il software Mobile Eyes, collegato al server centrale.

1. Introduzione

In questo capitolo viene presentato l'ambiente di lavoro utilizzato per effettuare gli esperimenti, la strumentazione necessaria e l'architettura software corredata di librerie.

1.1. Ambiente di lavoro

L'ambiente di lavoro utilizzato per effettuare gli esperimenti è stato il laboratorio di robotica. Questo è stato modellizzato tramite una scansione effettuata dai robot prima dello sviluppo dell'elaborato. Essendo l'obiettivo dell'elaborato quello di muovere più robot insieme, è stato necessario selezionare un ambiente di movimento sufficientemente ampio e, di conseguenza, robot meno ingombranti possibile.

1.2. Strumentazione utilizzata

Per eseguire l'esperimento sono stati utilizzati i robot Pioneer 1 Speedy e Tobor.

Entrambi i Pioneer hanno le medesime caratteristiche. Questi robot anolonomi sono dotati di tre ruote, due motrici fisse poste nella parte anteriore e una folle e pivottante nella parte posteriore. Per il movimento è applicato il meccanismo del differential drive. Tobor dal canto suo è equipaggiato anche con una pinza nella parte anteriore, però non utile ai fini dell'esperimento. I robot per orientarsi e per controllare la presenza di ostacoli fanno uso di un laser. Questo può rilevare da una distanza minima di 20 mm a una massima di 5600 mm con un angolo di scansione di 240°. L'errore è di 10 mm per le rilevazioni effettuate tra i 20 mm e i 1000 mm, e dell'1% per le rilevazioni a distanza superiore.

Per quanto riguarda il robot Speedy esso è attrezzato anche di un piattaforma di compensato posizionata nella sua parte posteriore, che ne aumenta le dimensioni rispetto a Tobor.

Per la verifica non è stato utilizzato Morgul a causa delle sue elevate dimensioni di ingombro, che avrebbero altrimenti costituito un ostacolo insormontabile per gli altri robot. Ciò viene anche accentuato dalle ristrette dimensioni dell'ambiente di lavoro.

Fig. 1 - Robot Pioneer 1 Speedy (sinistra) e Tobor (destra)

1.2.1. Librerie software

L'elaborato ha richiesto l'utilizzo delle seguenti librerie della MobileRobots:

1. Aria;
2. AriaForArnl;
3. ArNetworkingForArnl;
4. Arnl;
5. BaseArnl.

Esse sono state fondamentali per interfacciarsi ai robot e per poterli pilotare.

La libreria Aria rappresenta una SDK per accedere e gestire i robot di MobileRobots in modo efficiente. La gestione dei robot è disponibile con 2 approcci: l'invio di semplici comandi di manovra oppure tramite azioni.

L'accesso al robot invece è realizzato sfruttando la libreria di supporto ArNetworking: tale libreria utilizza un approccio client-server per implementare sia un'architettura di rete sia un protocollo per trasferire dati e comandi. ArNetworking utilizza un'architettura a 3 strati: un client esterno invia richieste ad un server presente sul calcolatore del robot; tale server poi si interfaccia con la libreria Aria per realizzare le richieste effettuate. Questa architettura prevede anche che un robot possa non solo essere server, ma anche client: per esempio in una situazione con più robot, un singolo nodo può richiedere agli altri informazioni come posizione corrente o rilevamenti dei laser.

La libreria Arnl ha invece come scopo la localizzazione del robot tramite l'utilizzo dei soli laser. Affinché la libreria possa essere utilizzata, è necessario soddisfare la sua dipendenza con BaseArnl. Tale libreria ha come scopo quello di offrire un'implementazione di path planning, nonché quello di offrire alcune classi base per la localizzazione. Oltre a ciò, BaseArnl contiene anche le classi principali per gestione di più robot contemporaneamente, sia in modalità peer to peer sia attraverso un coordinatore centrale. Affinché la comunicazione tra i vari robot funzioni correttamente, è necessario che esista un'architettura di rete e un protocollo di comunicazione; tali richieste sono soddisfatte dal pacchetto ArNetworkingForArnl, che rappresenta un sottoinsieme di ArNetworking. Infine, proprio come ArNetworkingForArnl, AriaForArnl rappresenta un sottoinsieme di Aria.

2. Il problema affrontato

Si consideri un ambiente in cui convivono più robot. Si vuole poter muovere questi robot contemporaneamente da uno stesso calcolatore, tramite un sistema che permetta la visualizzazione globale dello stato in cui ci si trova e quindi la sua manipolazione.

2.1. Sincronizzazione tra robot

Siccome si vuole realizzare un movimento contemporaneo nello stesso ambiente, è necessario sincronizzare i robot con un metodo specifico. Nel caso in oggetto la libreria ARNL offre due possibili approcci:

- Sincronizzazione tramite un unico server centrale.
- Sincronizzazione tramite peer to peer, cioè senza l'utilizzo di un coordinatore centrale per la gestione delle informazioni dei robot.

2.2. Visualizzazione e manipolazione dello stato

La presenza di più robot ha reso necessaria l'introduzione di un sistema unico per la visualizzazione dello stato e di conseguenza per la sua manipolazione. Nel concreto si tratta di visualizzare la mappa dell'ambiente e, al suo interno, la posizione corrente dei due robot, comprensiva delle rilevazioni dei loro sensori. Per quanto riguarda la manipolazione dello stato si è voluto un sistema che permettesse la guida automatica dei robot tramite pianificazione del percorso ottimo verso dei goal o opzionalmente anche la guida manuale.

3. La soluzione adottata

3.1. Architettura dei calcolatori

Il problema presentato ha richiesto, come già detto, un meccanismo di sincronizzazione, raggruppamento e visualizzazione delle informazioni dei robot. L'architettura della rete di funzionamento è strutturata da tre o più calcolatori connessi tra loro tramite una rete wireless Enterprise (la rete universitaria):

- Central server;
- Calcolatore MobileEyes;
- Calcolatori sui robot.

La connessione tra robot e central server avviene sulla porta 7272, che rimane aperta in modo che il central server possa acquisire le informazioni necessarie sulla posizione in cui si trovano i robot. A sua volta MobileEyes è connesso al central server; in questo modo riesce a rilevare i dati sui robot, quelli dei sensori e le informazioni sulle mappe utilizzate, nonché permettere all'utente di effettuare path planning o guida manuale. In altre parole il central server raggruppa e fornisce a MobileEyes i dati sui robot multipli, in modo che l'utente possa agire direttamente sui robot in modo trasparente e intuitivo, tramite l'interfaccia grafica dell'applicazione.

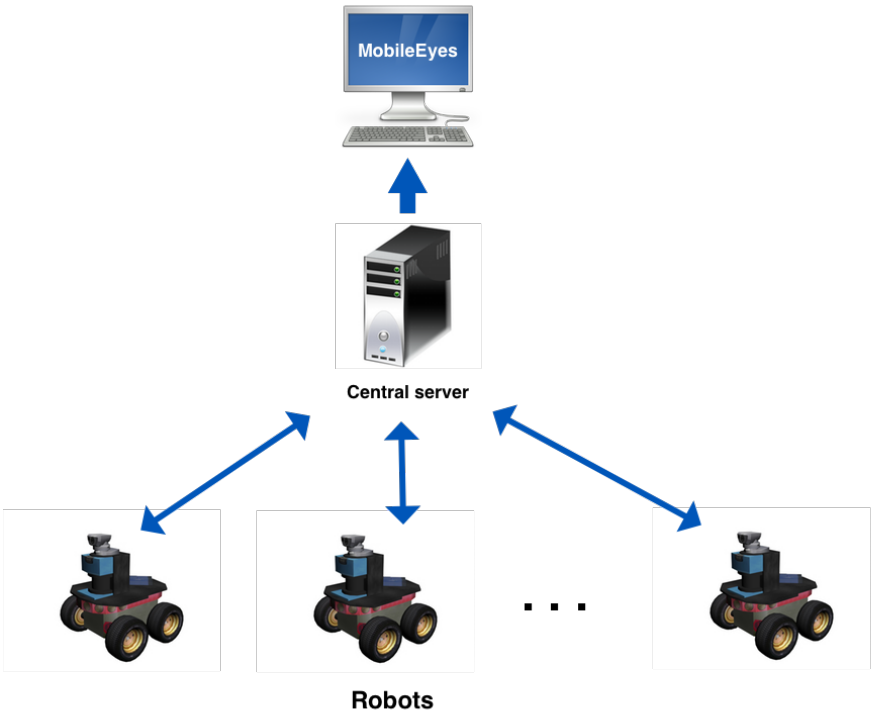

Fig. 2 - Architettura dei calcolatori

3.2. Central Server

Il programma sul central server permette di accettare connessioni sia da parte di un client (MobileEyes), sia da parte di altri server (i robot), in modo da poter creare una comunicazione diretta tra essi e permettere anche la condivisione d'informazioni tra i robot presenti nella mappa.

Il setup avviene tramite l'istanziazione di due coppie di classi.

- ArServerBase e ArSimpleOpener per rappresentare il server in ascolto dei robot e per effettuarne il setup. Questo server accetta connessioni dai robot presenti sulla porta specificata dal comando -<name>serverPort. In questo caso <name> è stato impostato a "robot" e la porta usata è la 5000, quella di default.
- ArServerBase e ArSimpleOpener per la gestione delle connessioni al client. Infatti questo server accetta connessioni da un client, in questo caso da parte di MobileEyes.

Dopodiché i due server sono aperti e avviati.

A questo punto sono creati degli oggetti per la gestione delle connessioni e delle funzionalità del server.

- ArCentralManager è un oggetto che tiene traccia dei robot che si connettono al central server. Per ogni connessione creata, è istanziato un oggetto ArCentralForwarder. Quando invece si connette un client, questo è informato su quali porte utilizzare per connettersi ai robot, tramite gli oggetti ArCentraForwarder creati in precedenza. Questi forwarder permettono quindi di inviare comandi ai robot.
- ArServerHandlerConfig è l'oggetto utilizzato per permettere al client di inviare le configurazioni ai robot o per richiederle.
- ArServerHandlerCommands e ArServerSimpleServerCommands, semplici strutture usate per inviare comandi al client.

Si effettua poi un setup utile per la raffigurazione dei dati sul client MobileEyes. In particolare questo passaggio è fatto tramite l'istanziazione della classe ArServerInfoDrawings e di ArCentralMultiRobot. Quest'ultima è in grado di tener traccia della posizione di ogni robot e di condividerla. Inoltre permette anche di inviare le informazioni sulla raffigurazione del sistema al client, tramite il passaggio dell'oggetto di tipo ArServerInfoDrawings al suo costruttore.

3.3. Codice sui robot

Il codice implementato sui robot può essere eseguito in due modalità:

- Modalità peer-to-peer, in cui i robot fanno da server per MobileEyes.
- Modalità central server, quella utilizzata in questo caso e che richiede la specifica dell'indirizzo IP del server tramite il comando "-centralServer".

Il programma, quando eseguito, carica i dati di configurazione del robot da un file particolare, solitamente chiamato <nome robot>.p. In questo file può anche essere specificata la mappa da utilizzare, che si presume ovviamente essere la stessa per tutti i robot attivi e connessi al central server. In caso invece non sia presente alcuna mappa, come nel caso in esame, andrà specificato il path del file mappa all'esecuzione del programma.

Nel programma vengono utilizzati oggetti e metodi specifici per effettuare il corretto setup del robot, in modo che i sensori e le comunicazioni via rete funzionino. Andando in ordine:

- Oggetto ArRobot che identifica il robot stesso.
- ArServerBase e ArSimpleOpener, oggetti per il setup del server sul robot.
- ArLaserConnector e ArSonarDevice per la configurazione e connessione rispettivamente dei laser e del sonar.
- ArMap, per la creazione e gestione della mappa.
- ArPathPlanningTask, oggetto per la gestione della pianificazione del percorso e che chiamerà particolari callback a seconda del verificarsi di particolari eventi (raggiungimento del goal, fail, ecc…).
- ArLocalizationTask per la localizzazione del robot, eseguita tramite l'algoritmo di Monte-Carlo basato sulle rilevazioni dei laser.

Dopo questa fase di setup, il server viene avviato e vengono creati oggetti per la gestione di robot multipli. Sono istanziate in particolare due classi:

- ArServerHandlerMultiRobot per la connessione al central server e l'invio di informazioni a esso. L'istanziazione di quest'oggetto richiede che siano specificati come parametri il robot stesso, l'oggetto server del robot associato al central server, la mappa e gli oggetti per la localizzazione e la pianificazione del path propri del robot.
- ArMultiRobotRangeDevice, oggetto per il recupero delle informazioni sugli altri robot dal central server. Questo è associato poi al robot e al path task.

Due ulteriori fasi compongono il programma.

La prima riguarda il perfezionamento del processo di pianificazione del percorso. Vengono aggiunti dei range device riguardanti la rilevazione agli infrarossi, quella tramite bumper e quella che fa riferimento a zone della mappa proibite. Inoltre è eseguito un metodo per il rallentamento del robot nel caso il punteggio di localizzazione sia basso e uno che lo ferma nel caso di perdita totale di localizzazione.

La seconda fase invece riguarda i servizi di localizzazione, o meglio di raffigurazione del robot tramite l'oggetto ArDrawingData. Questo permette infatti a MobileEyes di disegnare il robot stesso, i dati rilevati dai suoi sensori e un riferimento ai suoi contorni (un semplice quadrilatero attorno al robot).

È opportuno a questo punto spiegare l'utilità specifica delle due classe fondamentali utilizzate per gestire pianificazione e localizzazione.

3.3.1. ArPlanningPathTask

Questa classe rappresenta un task separato asincrono e adibito alla pianificazione del percorso che un robot deve seguire per raggiungere un dato goal. La pianificazione compie una ricerca basata su griglia per trovare il percorso più veloce dalla posizione del robot a un qualsiasi punto raggiungibile nell'ambiente d'azione. Dopodiché esegue le azioni sul robot per seguire il percorso pianificato. Contemporaneamente è eseguito il metodo "dynamic window" per evitare ostacoli dinamici, ossia non presenti nella mappa originale. Questo metodo consiste nell'esecuzione di due passi nel momento d'incontro con l'ostacolo:

1. Ricerca di uno search space valido.
2. Scelta della soluzione ottima nel search space, ristretta alle traiettorie circolari libere da ostacoli.

Richiedendo una localizzazione accurata del robot, l'utilizzo di tale classe rende necessario anche l'utilizzo di ArLocalizationTask.

Di seguito una visualizzazione dello schema ereditario di tale classe.

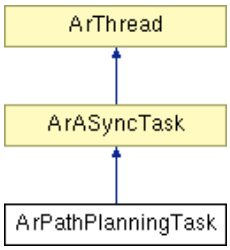

Fig. 3 - Schema ereditario della classe ArPathPlanningTask

3.3.2. ArLocalizationTask

Si tratta di una classe che effettua la localizzazione continua in un thread asincrono separato. L'algoritmo utilizzato è l'algoritmo Monte-Carlo, che sfrutta i laser per individuare la posizione del robot in una data mappa. L'informazione sulla localizzazione è sfruttata per la pianificazione del percorso e per la visualizzazione via MobileEyes.

Lo schema ereditario è simile al precedente.

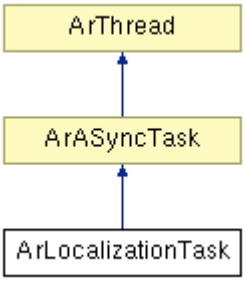

Fig. 4 - Schema ereditario della classe ArLocalizationTask

3.4. MobileEyes

MobileEyes (ME) è un software client per la visualizzazione del sistema robot-ambiente descritto finora tramite informazioni reperite da un server. Tale server può essere il robot stesso o, in questo caso, il central server a cui i robot sono collegati. All'apertura di ME viene infatti richiesto l'indirizzo IP del server a cui fare riferimento. Se si utilizza un robot verrà visualizzato solo quel robot, ma se si utilizza il central server, ogni robot a lui connesso verrà disegnato nella schermata di ME principale e all'interno della mappa richiesta. A questo punto è visualizzata a sinistra la lista dei robot disponibili. Selezionato il robot sarà possibile muoverlo manualmente con le apposite funzionalità, oppure indicare dei goal da raggiungere. Il robot in questo caso effettuerà una pianificazione del percorso e, in caso di ostacolo effettuerà nuovamente tale calcolo in modo da poterlo aggirare. Nel caso la pianificazione non vada a buon fine verrà invocata la callback relativa al fallimento nel raggiungere il goal. Viceversa, in caso di raggiungimento del goal, sarà eseguita la callback di successo. In entrambi i casi il robot si fermerà.

3.4.1. Mappe utilizzate

La mappa utilizzata per l'esperimento è stata ARLConGoal.map. Tuttavia, per testare più agevolmente il comportamento dei robot si è pensato di creare una seconda mappa identica alla prima, ma con goal differenti. In particolare si sono utilizzati 2 coppie di 4 goal, così da avere percorsi in ambienti più ampi e agevoli al movimento di robot multipli. Tale mappa è stata chiamata BBPARL.map ed è reperibile nella directory "/usr/local/Aria/maps/" sia su Tobor che su Speedy.

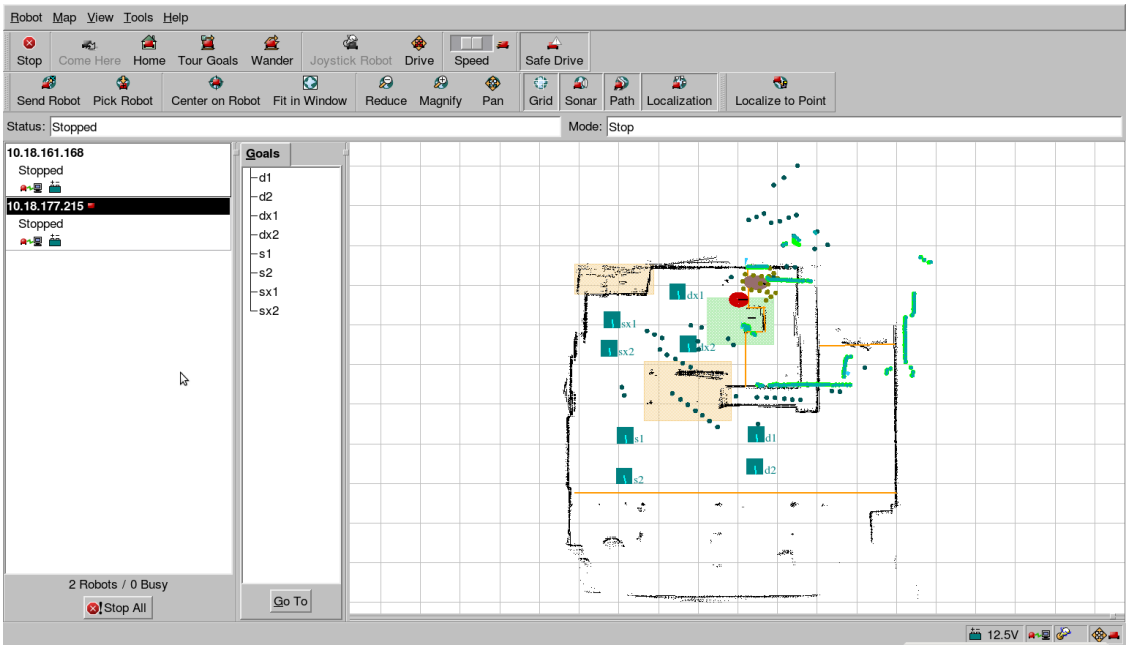

Fig. 5 - Screenshot MobileEyes, mappa BBPARL.map

4. Modalità operative

In questo capitolo si descrivono in modo dettagliato le modalità di installazione ed esecuzione del software per la manipolazione di robot multipli.

4.1. Componenti necessari

L'elaborato consiste in due programmi: BBPcentralServer.cpp e BBPmultiRobot.cpp. Il primo rappresenta il codice sorgente del server centrale mentre il secondo contiene le istruzioni che devono essere eseguite su ogni singolo robot. Il progetto è stato realizzato e testato utilizzando Tobor e Speedy, due robot Pioneer1 della MobileRobots. Nulla vieta che il programma possa essere eseguito su un qualsiasi robot in grado interpretare ed eseguire i comandi di ARIA e ARNL. È infatti necessario che queste due librerie siano installate sui calcolatori dei robot mobili e sul calcolatore che viene utilizzato come central server. Se non lo fossero, esse sono reperibili sul sito della MobileRobots. Infine è necessario che sui calcolatori dei robot e del central server sia disponibile la libreria stdc++6 versione almeno 4.4. Per la fase di creazione della mappa dell'ambiente è necessario installare il software MobileEyes.

Inoltre sono necessari i seguenti comandi sul central server: hostname 3.06 o superiori, curl, wget.

Sui robot invece deve essere presente i comandi dirname e find.

4.2. Modalità di installazione

Si suppone che i calcolatori a bordo dei robot abbiano già installate le librerie ARIA e ARNL. Per l'installazione del software MobileEyes, dopo averlo reperito dal sito della MobileRobots o dal sito del laboratorio di robotica avanzata, eseguire il seguente comando:

```
$ sudo dpkg –i <nomePacchetto>
```

Per l'esecuzione dei programmi BBPmultiRobot e BBPcentralServer non è necessaria nessuna installazione. In particolare BBPcentralServer deve essere trasferito sul calcolatore che viene utilizzato come server contrale in /usr/local/src/BaseArnl/examples mentre BBPmultiRobot deve essere trasferito sui calcolatori dei robot mobili in /usr/local/Arnl/examples.

4.3. Compilazione

Per quanto riguarda la compilazione, è stato inserito all'interno di ogni file tar.gz un makefile opportuno chiamato "Makefile". Questo contiene i comandi e i riferimenti alle librerie necessari per la compilazione dei file .cpp i cui eseguibili verranno chiamati in fase di esecuzione. In particolare il makefile permette di compilare sia il codice del central server che quello dei robot.

Per quanto riguarda il server centrale, la compilazione di BBPcentralServer.cpp ha reso necessaria l'esecuzione del comando g++ con i seguenti parametri specificati tramite macro nel makefile:

- ARNL_INCLUDES per specificare la locazione degli include necessari.
- SUPPORT_INCLUDES per il recupero della locazione in cui si trovani gli header file come Aria.h, ArNetworking.h e Arnl.h.
- SUPPORT_LIBRARIESPATH per il recupero del percorso relativo alle librerie dinamiche necessarie al central server come BaseArnl, ArNetworkingForArnl e AriaForArnl.
- ARNL_LIBRARIESPATH per la specifica del path delle librerie dinamiche arnl.
- LINKEROPTIONS per opzioni da passare al linker.

Per la corretta compilazione di BBPcentralServer è necessario che i pacchetti Arnl e Aria siano correttamente installati in /usr/local/Arnl e /usr/local/Aria. Nel caso in cui questo non sia vero è necessario specificare le macro ARNLDIRECTORY e/o ARIADIRECTORY all'esecuzione del makefile. Inoltre è possibile richiamare il comando make fornendogli la macro ARCHITECTURE: essa consente di compilare il programma con un'architettura diversa da quella presente sul sistema; per

esempio è possibile compilare il programma a 32 bit su un sistema a 64 bit settando ARCHITECTURE a m32.

Per esempio tramite il comando:

```
make ARNLDIRECTORY=/usr/my_local/my_Arnl
ARIADIRECTORY=/usr/my_local/my_Aria centralServer
```

È stato comunque inserito un help richiamabile tramite il comando `make help` che spiega l'utilizzo di tali macro nel caso i pacchetti siano in una locazione diversa.
Tramite il comando `make install` invece l'eseguibile risultante dalla compilazione (BBPcentralServer) e gli script utili all'esecuzione dell'esperimento (presentati successivamente) vengono installati nella directory /usr/local/bin come comandi eseguibili da console di comando.
Viceversa con il comando `make uninstall` vengono disinstallati.

Il file sorgente BBPMultiRobot.cpp deve essere compilato su ogni robot coinvolto nell'esperimento.
Allo stesso modo del central server, il makefile contiene riferimenti a librerie ben precisi e necessari alla corretta compilazione:

- ARNL_INCLUDES per la specifica degli include necessari.
- ARNL_LIBRARIESPATH per la specifica del path delle librerie dinamiche Arnl.
- LINKEROPTIONS per opzioni relative al linker.

Anche in questo caso il comando `make install` copia l'eseguibile risultante nella cartella /usr/local/bin come comando eseguibile. Il comando `make uninstall` lo rimuove.

Infine la mappa relativa all'esperimento viene copiata nella directory adibita alle mappe (di default /usr/local/Aria/maps).

Per una più chiara comprensione del codice e per completezza, di seguito vengono specificati i parametri di compilazione utilizzati all'interno delle macro presenti nel makefile:

- g++: nome del compilatore da utilizzare
- -g: produce l'eseguibile con delle informazioni di debugging
- -Wall: in fase di compilazione il compilatore mostra tutti i warning possibili
- -D: definisce una macro di compilazione
- -fno-exceptions: ignora le eccezioni di C++
- -fPIC: viene generato codice indipendente dalla posizione
- -I: indica una directory in cui cercare gli header files richiesti dal programma
- -L: indica una directory contenente le librerie dinamiche (-so)
- -l: importa una libreria dinamica (.so) presente in un folder di sistema oppure in una folder specificata con -L (senza il prefisso "lib" e senza l'estensione ".so")
- -o: come chiamare il file di output
- -Bstatic -Bdynamic: si cercherà di linkare la libreria passata a seguito di questo argomento staticamente/dinamicamente
- -Xlinker: passa un'opzione al linker

4.4. Script realizzati

4.4.1. RobotServerScript.sh

Questo script esegue sul robot mobile il programma BBPmultiRobot, in modo che possa instaurarsi una connessione tra il robot stesso e il server centrale.

Si noti che per potersi connettere al server centrale è necessario eseguire il programma BBPmultiRobot utilizzando il parametro "-centralServer" seguito dall'indirizzo IP del calcolatore su cui è eseguito il

programma BBPcentralServer. Inoltre non è richiesto il parametro “-cl”, in quanto il software stesso lo inserisce automaticamente.

Affinché lo script funzioni correttamente:

- Il central server deve essere già stato avviato.
- Lo script deve essere avviato specificando l'indirizzo IP del central server.
- La mappa BBPARL.map deve essere in /usr/local/Aria/maps.

Infine ogni programma deve essere chiamato sui robot coinvolti nell'esperimento, specificando lo stesso file .map (esecuzioni di BBPcentralServer con mappe diverse implicano la visualizzazione di più mappe all’interno di MobileEyes) e indirizzo IP del central server.

4.4.2. BBPAutologin.sh

Questo script permette di effettuare il login ai robot mobili utilizzati nell'esperimento.

Successivamente tramite ssh esegue sul robot in questione lo script RobotServerScript.sh passando come parametro l'indirizzo IP del server centrale.

4.4.3. MultiRobotInitialize.sh

Questo è lo script da lanciare per simulare l'esperimento.

Tale script esegue innanzitutto BBPcentralServer sul server centrale.

Successivamente due terminali eseguendo su ognuno di essi lo script BBPAutologin in modo da effettuare il login ai due robot mobili Tobor e Speedy.

Infine viene avviato il software MobileEyes connettendolo al calcolatore sul quale è stato avviato il server centrale, per poter monitorare il movimento contemporaneo dei due robot mobili. Nell'esperimento MobileEyes è avviato sul calcolatore che viene utilizzato dal server centrale e quindi è necessario inserire l'indirizzo 127.0.0.1 per effettuare la connessione. Prima di effettuare ciò, è consigliato attendere che i due robot mobili si siano connessi al central server.

4.4.4. MultiRobotFinalize.sh

Terminato l'esperimento è necessario killare i processi che sono stati precedentemente avviati sui robot mobili. Tale procedura è necessaria in quanto l'esecuzione del programma BBPmultiRobot, oltre a consumare memoria e CPU, blocca l'accesso alla comunicazione seriale con il robot stesso impedendo altre connessioni. Al termine dell'esperimento è quindi necessario eseguire lo script MultiRobotFinalizet.sh. Tale procedura esegue lo script AutoLoginToKillBBPMultiRobot.sh su entrambi i robot mobili utilizzati nell'esperimento, ovvero Speedy e Tobor.

4.4.5. AutoLoginToKillBBPMultiRobot.sh

Questo script permette di effettuare il login ai robot mobili (Tobor e Speedy) utilizzati nell'esperimento.

Successivamente tramite ssh esegue sul robot in questione lo script KillBBPmultiRobot.sh.

4.4.6. KillBBPmultiRobot.sh

Tale script effettua un kill del processo BBPmultiRobot sul robot. In particolare esegue:

```
kill -s 9 $(ps -e | grep BBPmultiRobot | tr -s " " | sed 's/^[ \t]*//'
|cut -d" " -f1)
```

4.5. Flusso di esecuzione dell'esperimento

Scompattare i file tar.gz nelle locazioni desiderate sul central server e sui robot.

Compilare i codici sorgenti tramite i makefile specifici, eseguendo i seguenti comandi.

- `make`
- `make install`

Eseguire il comando derivante dallo script omonimo MultiRobotInitialize sul central server.

In seguito a tale esecuzione verranno avviati:

- programma MobileEyes;
- tre terminali relativi al central server e ai due robot mobili Speedy e Tobor utilizzati nell'esperimento.

Attendere l'avvio del central server e successivamente inserire le password per connettersi ai due robot mobili. Infine connettersi al server centrale con il software MobileEyes e simulare il movimento contemporanea dei due robot.

Concluso l'esperimento è necessario chiudere i tre terminali sopra citati e killare i processi avviati precedentemente sui robot mobili eseguendo lo script MultiRobotFinalizeScript.sh.

4.6. Ambiente di test

L'esperimento è stato condotto utilizzando le seguenti versioni del dato software:

	Central Server	**Speedy**	**Tobor**
SO	Linux 3.5.0-51-generic (Ubuntu). Architettura: x86_64, Linux 3.2.0-4-amd64 (Debian 3.2.57-3) Architettura: x86_64	Linux 2.6.32-5-686 Architettura: i686	Linux 2.6.32-5-686 Architettura: i686
g++	4.6.3	4.4.5	4.4.5
ARIA	2.8.1	2.8.1	2.8.1
ARNL	1.8.1	1.8.1	1.8.1
BASEARNL	1.8.1	1.8.1	1.8.1
readlink	8.13		
dirname	8.13		
hostname	3.06		
curl	7.26.0		

5. Conclusioni e sviluppi futuri

Il lavoro svolto ha permesso di instaurare una comunicazione tra due o più robot mobili attraverso un central server. In questo modo è possibile, per un utente, visualizzare le informazioni su di essi ed eseguire comandi attraverso un software client come MobileEyes. In particolare è possibile muovere contemporaneamente i robot mobili nell'ambiente di esecuzione tramite la pianificazione di un percorso verso i goal.

Questo elaborato può essere una base valida per sviluppi futuri riguardanti sistemi multirobot quali ad esempio:

- Utilizzo di più robot mobili in modo cooperativo per effettuare una più rapida mappatura di un ambiente sconosciuto.

- il movimento contemporaneo di più robot con traiettorie incidenti. In caso di deadlock da parte di due robot, una possibile soluzione potrebbe essere quella di fermare uno dei due robot ed effettuare un wander sul secondo, per poi riattivare il robot precedentemente fermato.

Bibliografia

[1] Aria Developer's Reference Manual:

http://www.ing.unibs.it/~arl/docs/documentation/Aria%20documentation/Current/Aria%20docs/

[2] ARNL Documentation:

http://www.ing.unibs.it/~arl/docs/documentation/Aria%20documentation/Current/Arnl%20docs/ARNL-Reference/

[3] BaseArnl Documentation:

http://www.ing.unibs.it/~arl/docs/documentation/Aria%20documentation/Current/Arnl%20docs/BaseArnl-Reference/

[4] Manuale g++:

https://gcc.gnu.org/onlinedocs/gcc-4.9.0/gcc/

[5] Cassinis, R. Lucidi lezione 19:

http://www.ing.unibs.it/~cassinis/Dida/current/roba/lezioni/Lezione%20RA%2019.pdf

Indice