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)

Sistema di telecomando via applet di Speedy

Elaborato di esame di:

Alberto Cima, Nicola Modonesi

Consegnato il: **09 luglio 2007**

Sommario

In questo lavoro di progetto è stato realizzato ed implementato un sistema di telecomando remoto via applet di Speedy Questo telecomando appare all'utente come costituito da un'interfaccia grafica che permette di comandare i movimenti del robot Speedy sotto la supervisione del suo sistema di anticollisione in condizioni standard e senza questa supervisione nelle circostanze ove è necessario avvicinarsi a particolari oggetti come per esempio la docking station di ricarica del robot.

1. Introduzione

L'obbiettivo del lavoro svolto è fornire un interfaccia grafica accessibile dalla rete che consenta di guidare manualmente il robot Speedy. L'interfaccia grafica è stata realizzata tramite Applet, ed è costituita da un joystick virtuale, che consente di regolare la velocità di avanzamento e rotazione del robot, e da un controllo che permette di attivare e disattivare i comportamenti per il controllo automatico delle collisioni. Sul robot viene attivato tramite una CGI il server di guida del robot, quindi tramite un opportuno protocollo viene permessa la comunicazione con la applet e quindi l'effettivo controllo remoto di Speedy

2. Il problema affrontato

L'obbiettivo del lavoro svolto è fornire un sistema di guida per il robot Speedy che possa essere eseguito da browser e che sia accessibile dalla rete interna al laboratorio e da internet. Il sistema di guida deve garantire le seguenti funzionalità:

- Connessione automatica senza bisogno di inserire alcun parametro o comando
- Possibilità di regolare la velocità di avanzamento e di rotazione del robot tramite un joystick virtuale
- Possibilità di attivare e disattivare la funzione *safe drive* per il controllo automatico delle collisioni
- Possibilità di attivare e disattivare il server presente sul calcolatore di Speedy

Il lavoro è stato suddiviso in quattro parti logicamente distinte, il controllo del server di guida (*remote*), la realizzazione dell'interfaccia grafica della applet e della pagina html, la realizzazione del protocollo di collegamento e la realizzazione del server di controllo remoto per l'elaborazione dei dati passati via rete dalla applet.

2.1. Controllo del server di guida (*remote*)

Per il controllo del server di guida dal nome *remote* è stato pensato di implementare una cgi che lo mettesse in moto e una cgi che lo spegnesse, a questo scopo è stato quindi necessario avere un server web sempre attivo.

2.2. Interfaccia grafica

Per permettere all'utente di accendere e spegnere il server *remote* si è pensato di implementare un form html che richiamasse le cgi citate al punto precedente.

Per permettere all'utente di controllare il funzionamento del robot sono stati creati il joystick grafico e una checbox per l'abilitazione/disabilitazione del *safe drive* (impostazione di una diversa scala di priorità dei comportamenti implementati in *remote*) tramite una applet.

2.3. Protocollo

Il protocollo studiato doveva permettere il passaggio una terna di interi, uno per la velocità di avanzamento, uno per la velocità di rotazione ed uno con la priorità da dare al comportamento associato al movimento imposto dall'utente.

2.4. Server di guida

A questo punto si è dovuto creare un server di guida (*remote)* affinché gestisse i tre ingressi consentendo la guida del robot nelle due modalità previste.

3. La soluzione adottata

3.1. Struttura dei due sistemi client-server

La struttura del software di controllo remoto è una tipica struttura client-server. Esistono due sistemi client server nel nostro progetto, il primo è quello costituito dal browser web e dal server web Apache, il secondo è quello costituito dalla applet Java e dal server di guida *remote.*

Il primo funziona con i protocolli standard dell'html ed il secondo funziona con un protocollo implementato ad hoc. In particolare la parte client è rappresentata dalla applet scritta in Java a differenza del server di guida chiamato *remote* che è stata implementato in C.

3.2. Interazione dei due sistemi client-server

L'interazione dei due sistemi client-server è stata così pensata:

1. su Speedy è sempre attivo il web servere *Apache*
2. da browser web ci si collega al server web scaricando la pagina html che contiene un form html e la applet
3. tramite il form html è possibile eseguire le CGI di accensione e spegnimento del sever di guida *remote*
4. tramite la applet è invece possibile comunicare con il server *remote* e quindi guidare il robot.

3.3. Implementazione delle CGI

La CGI che accende il server di guida *remote* è stata implementata con uno script in bash, la cui istruzione fondamentale è questa:

```
/usr/local/Aria/cimodo/remote –rp /dev/ttyUSB0
```

che fa partire il programma remote passando il parametro /dev/ttyUSB0 che rappresenta la porta di connessione tra il computer su cui viene eseguito il server di guida e i servo comandi del robot.

A questo scopo è stato necessario dare a qualsiasi utente il privilegio di scrittura e lettura sul dispositivo /dev/ttyUSB0 tramite l'istruzione da riga di comando

```
sudo chmod 666 /dev/ttyUSB0
```

Per quanto riguarda l'implementazione della CGI per lo spegnimento del server di guida è stata realizzata anch'essa in bash, l'istruzione fondamentale in questo caso è:

```
killall -9 remote
```

questa istruzione invia il messaggio di eliminazione del processo *remote* (creato in fase di esecuzione del programma *remote*) e dei suoi processi-figlio.

3.4. Interfaccia grafica

L'interfaccia grafica è costituita da 3 componenti principali: il joystick virtuale, un pulsante per abilitare/disabilitare la modalità di guida *safe drive* e un feedback testuale che mostra a video i valori della velocità di avanzamento e rotazione inviati al robot.

3.4.1. Joystick virtuale

Il joystick virtuale è realizzato tramite un cerchio blu che puo' essere spostato trascinandolo con il mouse; se il pulsante del mouse viene rilasciato il joystick torna in posizione neutra. Spostando il cerchio sull'asse verticale si modifica la velocità di avanzamento del robot, mentre spostandolo sull'asse orizzontale si modifica la velocità di rotazione; quando il cusrsore è nel centro il valore della velocità trasmessa è zero sia per l'avanzamento che per la rotazione.

Uno screenshot della Applet è mostrato in figura.

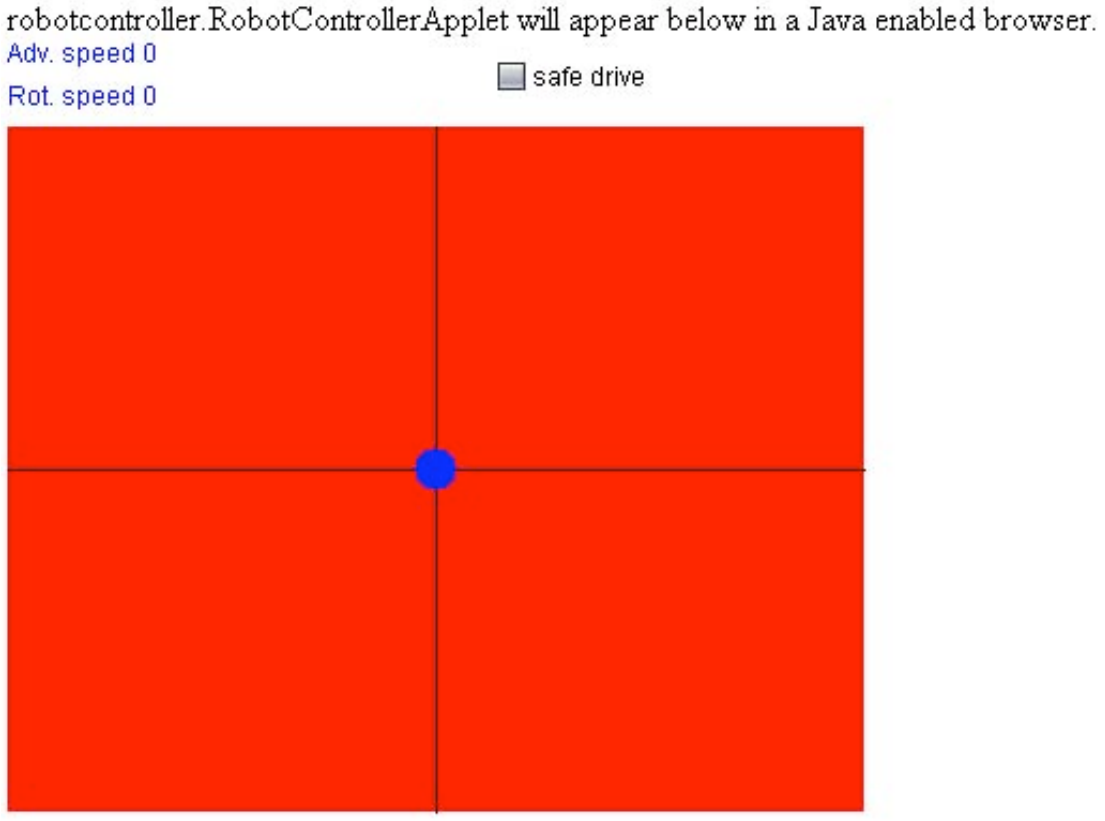

Figura 1

Le velocità di avanzamento e di rotazione sono calcolate dalla applet moltiplicando la distanza tra il punto blu e il centro degli assi tramite una opportuna costante in modo da raffinare il sistema di guida.

3.4.2. Modalità di guida *safe-drive*

Per poter muovere il robot liberamente vicino ad ostacoli e per poterlo quindi riposizionare nella sua stazione di ricarica è stato necessario inserire la possibilità di riordinare le priorità dei comportamenti del robot; nella applet è stato aggiunto una checkbox denominata *safe drive* per tale scopo: quando il *safe*

drive è abilitato allora il comportamento anticollisione ha priorità sui comandi di guida dell'utente, quando il *safe drive* è disabilitato allora i comandi di guida dell'utente hanno priorità sul comportamento di anticollisione.

3.4.3. Collegamento (della applet al remote)

Per guidare il robot si è scelto di inviare i 2 valori di velocità (di avanzamento e di rotazione) e un terzo per il controllo della modalità di guida.

Al fine di controllare la priorità di esecuzione dei comportamenti sul robot è usato il terzo intero trasmesso tramite il protocollo. Questo intero è stato salvato nella variabile intera dal nome *weight* che viene inizializzata al valore 25 (valore con cui il comportamento anticollisione ha priorità sui comandi di guida dell'utente). Il valore di questa variabile viene cambiato tramite l'evento provocato dalla spunta della checkbox associata secondo la seguente logica:

```
if  checked
then weight = 25
else weight = 120
```

Di seguito viene riportato la versione integrale del codice di gestione dell'evento:

```
public boolean action(Event e, Object arg) {
  System.out.println("Action chiamato: \u008F successo qualcosa");
  if (e.target == selezione) {
    if (selezione.getState()) {
      labelSelezione.setText("ON");
      weight = 25;
    }
    else {
      labelSelezione.setText("OFF");
      weight = 120;
    }
  }
```

3.5. Implementazione del protocollo parte client

La applet invia al sever *remote* 3 numeri interi separati da uno spazio; il primo è la velocità di avanzamento, il secondo è la velocità di rotazione e il terzo è un intero che rappresenta la priorità con la quale il comportamento di guida manuale viene eseguito rispetto al sistema di anti-collisione (25 safe *drive* attivato, 120 disattivato).

Nella parte client i punti salienti della implementazione del protocollo sono due, il collegamento al server e l'invio dei dati sul canale.

Il collegamento al server *remote* è stato implementato tramite il seguente metodo (*connect()*) della applet Java del quale viene qui riportato una versione semplificata:

```
private void connect() {
...
      String hostName;
```

```
      int serverPort;
      serverAddress = InetAddress.getByName(hostName);
      if ( (serverPort = Integer.parseInt(getParameter("serverPort")))
<1024) {
        System.out.println("Error - no port defined");
      }

      socket = new Socket(serverAddress, serverPort);
      in =
          new BufferedReader(
              new InputStreamReader(
                  socket.getInputStream()));
      out =
          new PrintWriter(
              new BufferedWriter(
                  new OutputStreamWriter(
                      socket.getOutputStream())), true);
    }
    ...
```

Come si vede dal codice il metodo colleziona le informazioni per la creazione del socket: indirizzo al quale si trova il server remote, la porta alla quale risponde. Per quanto riguarda la parte di invio dei dati sul canale l'implementazione è rappresentata dal metodo *trasmi()t* che riportiamo in forma semplificata qui di seguito:

```
private void transmit() {

    ...

    out.println(vel + " " + rotVel + " " + weight);

    String str = null;

  }
```

Come si vede dal codice vengono "stampati" sul canale i tre valori di velocità di avanzamento, velocità di rotazione e priorità del comportamento ad essi associato.

Certe versioni della *Java Virtual Machine* per questioni di sicurezza bloccano come impostazione di default la connessione ad indirizzi pubblici, in questi casi bisogna per permettere alla applet di connettersi al server *remote* bisogna aggiungere la seguente istruzione nella sezione *grant* del file *java.policy* che si trova nella sotto-directory *security* nella porzione di file-system dedicata all'installazione della JVM (per esempio C:\Program Files\Java\jre1.5.0_06\lib\security):

```
grant {

      ...

      permission java.net.SocketPermission "192.167.22.93:8103", "connect,resolve";
```

```
};
```

3.6. Implementazione del protocollo parte server

Nel programma *remote* vengono prima definiti i costruttori e le funzioni con le quali cambiare e settare le velocità di avanzamento e rotazione del robot, successivamente vengono definite le strutture per l'utilizzo dei sonar. Nel *main* vengono gestiti dapprima gli argomenti passati al momento dell'esecuzione del programma, successivamente vengono inizializzate le Action di gestione dei comportamenti del robot, vengono poi definite e inizializzate le variabili che verranno associate ai parametri passati tramite il protocollo. A questo punto vengono associati i comportamenti al robot, ciascuno con la propria priorità.

```
robot.addAction(&recover, 100);
robot.addAction(&avoidFrontNear, 50);
robot.addAction(&avoidFrontFar, 49);
robot.addAction(&VarSpeed, 25);
```

A questo punto, dopo aver fatto partire il robot con l'istruzione

```
robot.runAsync(true);
```

viene la parte che gestisce il protocollo: tramite il socket il programma riceve in ingresso tre numeri interi separati da uno spazio e che interpreta come velocità di avanzamento, velocità di rotazione e priorità del comportamento ad esse associato. Viene quindi rimossa l'azione precedentemente associata al controllo manuale del robot (VarSpeed) e subito reinizializzata con il nuovo peso associato.

```
while (1)
  {
        mySocket->writeString("Non mi rompere le scatole!");
        receivedBuffer=mySocket->readString();
        printf ("Ho ricevuto la stringa %s\n",receivedBuffer);

        sscanf(receivedBuffer,"%d %d %d",&newSpeed,&newRotSpeed,
&weight);
        if (newSpeed < -900) break;
        VarSpeed.setSpeed(newSpeed,newRotSpeed);
        printf("weight=%d",weight);
        robot.remAction(&VarSpeed);
        robot.addAction(&VarSpeed, weight);
        mySocket->writeString(receivedBuffer);
  }
```

4. Modalità operative

4.1. Componenti necessari

Per fare funzionare il sistema realizzato servono i seguenti componenti:

1. il robot Speedy

2. il server Apache su Speedy
3. le cgi startserver.cgi e stopserver.cgi per far partire e fermare il server *remote*
4. il server *remote* scritto in C per comandare il robot
5. il *file RobotControllerApplet.html* e la relativa applet *RobotControllerApplet.class* per guidare il robot

Per utilizzare il sistema realizzato bisogna:

1. aver acceso i motori su Speedy
2. Speedy sia correttamente alimentato
3. utilizzare un browser internet con Java abilitato per connettersi alla pagina *RobotControllerApplet.html:*

   a questo indirizzo *http://192.167.22.93:8082/~remote/RobotControllerApplet.html*
4. aver acceso il server remote dalla pagina *RobotControllerApplet.html*
5. avere garantito l'accesso della applet alla rete, fare riferimento al punto 3.5 di questa relazione
6. essere sicuri che l'indirizzo e la porta alla quale risponde il server siano validi

4.2. Modalità di installazione

Per installare il sistema realizzato bisogna fare come segue:

1. installare il web-server Apache su Speedy e configurarlo perché abbia una directory in cui posizionare gli script CGI (/cgi-bin)
2. mettere in questa directory le due CGI realizzate: startserver.cgi, stopserver.cgi. Successivamente è necessario garantire i privilegi di esecuzione di questi scritp all'utente che esegue il server Apache. Attenzione: le CGI sono scritte in bash, occorre quindi un sistema operativo in grado di eseguirle
3. installare in una directory a scelta dell'utente il server *remote*, nel nostro caso questa directory è /usr/local/Aria/cimodo, successivamente dare il privilegi di esecuzione all'utente che esegue Apache.
4. garantire all'utente che esegue le cgi di leggere/scrivere dalla porta usb che connette il computer di Speedy ai servo dei motori. Fare riferimento al punto 3.3 di questa relazione
5. assicurarsi che il codice delle CGI punti correttamente al file *remote* in dipendenza della directory di installazione di questo file
6. localizzare una directory dove Apache può fornire ai web-browser i file contenuti all'interno (nel nostro caso /home/remote/public_html) e posizionare il file *RobotControllerApplet.html*, creare poi una sotto-directory e chiamarla robotcontroller, in questa direcotory bisogna quindi posizionare il file *RobotControllerApplet.class*
7. controllare la validità dei link alle CGI nel file *RobotControllerApplet.html* in dipendenza dal percorso delle directory dove si trovano il file html e le CGI.

4.3. Avvertenze

Eventuali punti critici per l'installazione e l'esecuzione del sistema realizzato sono tracciabili ai punti:

- i punti 4.1 e 4.2 per componenti necessari e modalità di utilizzazione/installazione
- 3.3 per i privilegi di scritture/lettura sulla porta usb
- 3.5 per l'esecuzione della applet da browser web

5. Conclusioni e sviluppi futuri

La progettazione di questo sistema ci ha permesso di interagire con un robot mobile quale Speedy, in particolare ci ha permesso di notare come la programmazione orientata ai comportamenti sia necessaria nel campo della robotica mobile. Questo lavoro ci ha inoltre dato la possibilità di toccare con mano una serie di problematiche contingenti legate al funzionamento del robot, problematiche che abbiamo affrontato con entusiasmo e curiosità e che ci sentiamo di aver risolto con una buona efficacia.

Il sistema progettato potrebbe essere migliorato per quanto riguarda l'efficienza, andando a rivedere certe scelte progettuali come la modalità d'interazione dei due sistemi clint-server utilizzati oppure l'ottimizzazione del codice di *remote*.

Il sistema progettato potrebbe essere esteso da nuove funzionalità:

- la visualizzazione dell'immagine fornita da una webcam per la visualizzazione di Speedy e dei suoi movimenti
- un indicatore dello stato di accensione dei motori di Speedy
- un indicatore dello stato di esecuzione del server *remote*
- un indicatore del livello di batteria di Speedy

Bibliografia

Nessuna Bibliografia

Indice