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

Laboratorio di Robotica Avanzata
Advanced Robotics Laboratory

Corso di Robotica
(Prof. Riccardo Cassinis)

Controllo remoto di un robot ActiveMedia con un terminale Android

Elaborato di esame di:

Emanuele Barbeno, Marco Filippini, Marco Montagna

Consegnato il: **17 giugno 2013**

Sommario

Il lavoro svolto consiste nella realizzazione un'applicazione per Android che permetta di governare manualmente un qualunque robot ActiveMedia, attraverso l'uso dell'accelerometro. In questo modo si rende più semplice e intuitiva la guida di un robot mobile.

1. Introduzione

Il progetto che verrà ora esposto nasce nell'ambito del corso di "Robotica Industriale" e il suo scopo principale consiste nell'implementare un sistema di controllo, complementare a quelli già esistenti, che permetta di governare i robot mobili (Active Media) presenti nel laboratorio di robotica dell'Università degli Studi di Brescia.

Gli attuali sistemi di controllo remoto dei robot mobili, impiegati nel laboratorio di robotica, si basano sull'utilizzo di programmi di guida installati sull'elaboratore del robot, oppure sull'impiego di un'applicazione [Lopez 2012] compatibile con il sistema operativo iOS, e quindi sviluppata per dispositivi mobili Apple. Il presente progetto intende sviluppare un infrastruttura che permetta la guida del robot tramite un applicazione compatibile con un dispositivo dotato di sistema operativo Android.

2. Il problema affrontato

Il presente elaborato si pone l'obiettivo di realizzare un'infrastruttura client-server, che permetta la guida di un robot ActiveMedia attraverso un terminale dotato di sistema operativo Android.

Il terminale, che realizza la parte client, deve permettere l'interazione remota tra l'utente utilizzatore e il robot attraverso l'orientamento del dispositivo stesso. A seconda dell'inclinazione imposta al dispositivo, l'applicazione client invierà all'applicazione server dei dati, attraverso i quali l'applicazione server ricaverà i comandi da impartire ai motori del robot, in modo da variarne la velocità.

Nel presente elaborato la parte client verrà implementata attraverso un applicazione Android compatibile con un tablet Acer Iconia 500, dotato di sistema operativo Android 3.2.

L'applicazione server, che dovrà essere installata sul laptop montato a bordo del robot, dovrà essere in grado di ricevere i comandi provenienti dal client, elaborarli correttamente generando istruzioni comprensibili per il robot e attraverso queste istruzioni comandare i motori del robot per realizzare il movimento richiesto (che può essere un movimento in avanti, all'indietro oppure una rotazione in senso orario o antiorario).

Nel presente elaborato la parte server verrà implementata attraverso un programma scritto in linguaggio C++, compilato e installato su un laptop dotato di sistema operativo Debian 7.

2.1. Gli strumenti necessari

Per poter rilevare correttamente l'inclinazione imposta, il dispositivo Android utilizzato deve essere dotato di sensori in grado di rilevare la sua posizione, relativamente ad una posizione di partenza. Client e server devono poter implementare un canale comune per realizzare la loro comunicazione. Le due applicazioni sono realizzate utilizzando due linguaggi diversi (C++ per la parte server, Java per la parte client) e sono eseguiti su sistemi operativi diversi (Debian 7 per la parte server e Android 3.2 per la parte client) e quindi il canale di comunicazione deve essere scelto opportunamente, affinchè sia realizzabile su entrambe le piattaforme.

La parte server deve poter essere in grado di comunicare con il robot. Vi devono essere cioè, sia a livello implementativo che a livello fisico, dei canali attraverso i quali l'applicazione server riesca a comunicare direttamente con l'interfaccia del robot.

Infine, entrambi i componenti (server e client) devono poter accedere ad una rete wi-fi comune oppure riuscire a comunicare attraverso una connessione ad internet.

2.2. Breve descrizione degli ambienti di lavoro

2.2.1. Lato Server

L'applicazione server è installata su un laptop dotato di sistema operativo Debian 7. Debian è un sistema operativo open source, basato su kernel linux, corredato (all'installazione) unicamente con software libero (la maggior parte del quale rilasciato con licenza GNU/GPL). Il suo sviluppo è tenuto da un'associazione chiamata Debian Project, il cui intento è realizzare un sistema operativo completamente gratuito. Una caratteristica fondamentale del sistema operativo Debian, è rappresentata dalla stabilità dell'intero sistema. Questa caratteristica è dovuta in parte alla struttura del sistema operativo stesso, e in parte ai severi controlli a cui ogni nuovo software viene sottoposto, prima di essere introdotto in una nuova versione del sistema operativo. Ad un alto livello di astrazione, è possibile definire la struttura del sistema operativo Debian attraverso due livelli:

1. al primo livello, quello più basso, si trova il kernel che rappresenta il cuore del sistema operativo, incaricato di realizzare la comunicazione tra programmi applicativi e risorse hardware;
2. al secondo livello si trovano i programmi applicativi (di sistema, utente, …), le interfacce e le librerie condivise.

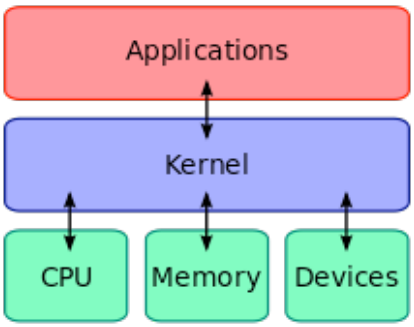

Figura 1: Struttura del Sistema Operativo
(source: https://en.wikipedia.org/wiki/Operative_system)

Debian dispone di numerosi tool per la scrittura e compilazione di nuovi programmi. In particolare nell'installazione di default è possibile trovare il compilatore GCC, necessario per la compilazione di programmi scritti in linguaggio C e C++. L'applicazione server del presente progetto, che verrà realizzata in linguaggio C++, si andrà a collocare nel secondo strato del sistema operativo.

2.2.2. Lato Client

L'applicazione client dovrà essere installata su un dispositivo dotato di sistema operativo Android. Android è un sistema operativo open source, attualmente sviluppato da Google, il cui obiettivo è realizzare la portabilità del sistema operativo stesso su diversi dispositivi, quali smartphone, tablet, televisori e altri dispositivi dotati di un microprocessore.

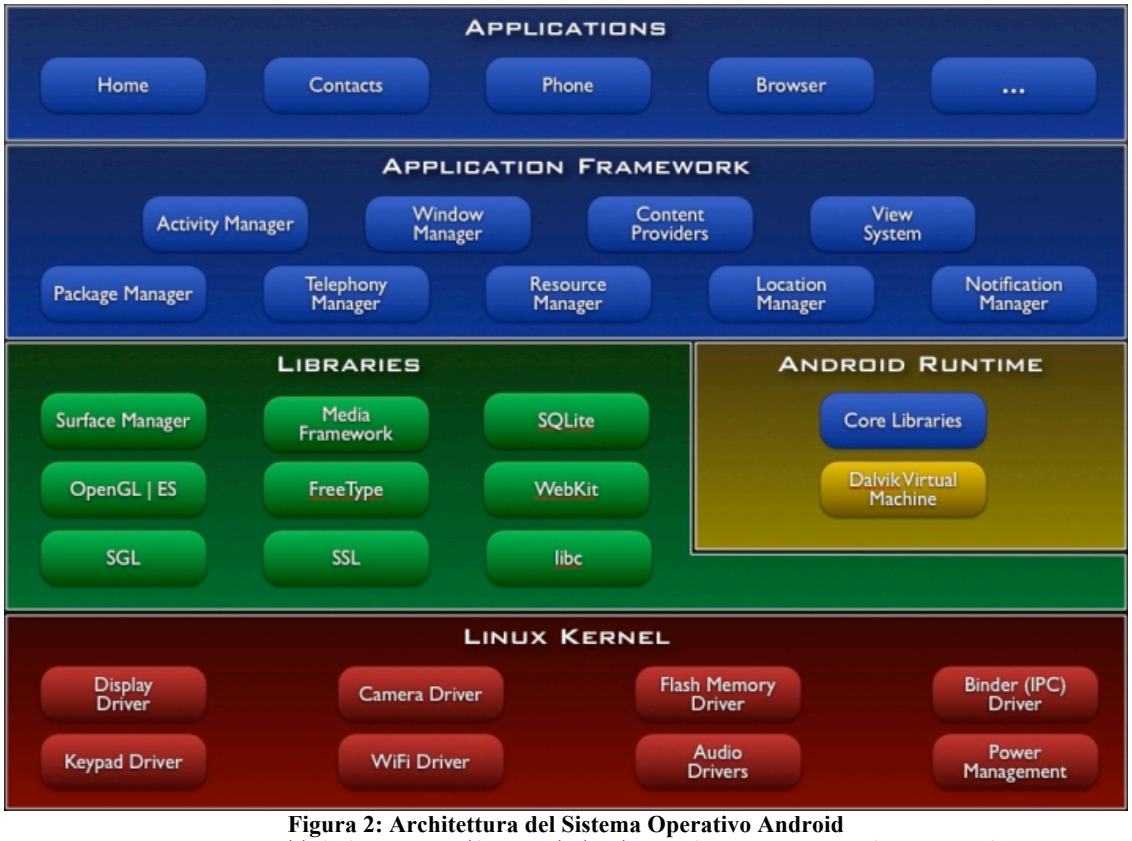

Figura 2: Architettura del Sistema Operativo Android
(source: http://wiki.dave.eu/images/c/c2/Android-system-architecture.jpg)

Il sistema operativo è stratificato in 4 livelli:

1. al primo livello si trova il kernel linux (v.2.6 fino ad Android 4.0, v.3.0 dalla Android 4.0 in poi);
2. al secondo livello si trovano le librerie di sistema e le API, entrambe scritte in linguaggio C;
3. al terzo livello si trova il framework per le applicazioni, che mette a disposizione delle app android le funzionalità del sistema operativo;
4. al quarto e ultimo livello si trovano le applicazioni installate nel sistema operativo.

Le applicazioni per un sistema operativo Android sono scritte in linguaggio Java, dalla cui compilazione si ottengono file scritti in linguaggio bytecode. Il sistema operativo Android non esegue direttamente questo codice, ma lo trasforma in un formato chiamato dex-code I file dex risultanti saranno poi eseguiti dalla Dalvik Virtual Machine, ovvero la macchina virtuale che Android utilizza per gestire le applicazioni installate nel sistema. Il codice dex e la Dalvik Virtual Machine, sono stati appositamente studiati e ottimizzati per dispositivi con risorse limitate (quali ad esempio tablet e smartphone). La Dalvik Virtual Machine si trova al livello II, al pari di librerie e API di sistema.

3. La soluzione adottata

Per poter comandare un robot tramite l'accelerometro si sono utilizzati due componenti distinte: un server (scritto in C++) installato sul calcolatore attaccato al robot che riceve i comandi inviati da un client (applicazione Android) risiedente su un tablet (nel nostro caso un Acer ICONIA A500).

3.1. Server

Il server è formato da 5 file:

- main.cpp: main del programma che permette di eseguire il server;
- Server.h: definizione dei metodi della classe Server che permette l'inizializzazione del robot, l'inizializzazione del socket e la gestione della comunicazione con il client;
- Server.cpp: implementazione dei metodi descritti nel suo file header;
- VarSpeedAction.h: definizione dei metodi della classe VarSpeedAction che permette di settare la velocità del robot;
- VarSpeedAction.cpp: implementazione dei metodi descritti nel suo file header.

È un server monothread, cioè non permette il collegamento simultaneo di due client ma riesce a gestirne solamente uno alla volta. Per la comunicazione con il robot si è deciso di utilizzare la libreria Aria, scritta in linguaggio C++, che mette a disposizione del programmatore funzioni in grado di interagire con robot ActiveMedia.

➢ **Di default il server accetta connessioni sulla porta 8080.**

3.2. Client Android

Il client android risulta essere diviso in 4 package distinti:

- *main*: contiene l'activity principale che chiede i parametri del server all'utente e gestisce l'apertura del collegamento tramite socket. Al suo interno è presente solo una classe "MainActivity" che gestisce l'interfaccia utente "dell'activity principale";
- *control*: contiene le classi che permettono di comunicare col server (inviare e ricevere parametri) e visualizzare le informazioni ricevute. Al suo interno sono presenti quattro classi:
  - *HotPotatoes*: che permette di controllare l'interfaccia utente "dell'activity di controllo" e gestisce i thread che permettono lo scambio dei dati con il server;
  - *BouncingBallView*: listener che trasforma i valori letti dal sensore nella forma pronta per l'invio sul socket (cioè un numero con la virgola compreso tra -1 e 1). Inoltre disegna una pallina di colore verde su un radar nero, posizionandola secondo i valori letti dal sensore;
  - *ReceiveThreadSocket*: si occupa di ricevere i dati dal server tramite il socket e capisce quando la connessione è caduta, notificandolo a HotPotatoes;
  - *SendThreadSocket*: si occupa di inviare i valori presi da BouncingBallView al server utilizzando il socket;
- *settings*: contiene le classi che permettono di modificare le impostazioni dell'applicazione, in questo caso la scelta dei parametri del robot da visualizzare e la cancellazione delle connessioni recenti. Al suo interno sono presenti due classi:
  - *SettingsActivity*: che gestisce l'interazione con "l'activity di controllo", cambiando il layout da visualizzare nel GenericFragmentSettings;
  - *GenericFragmentSettings*: mostra i dettagli della voce del menu scelto nella SettingsActivity in modo da poter modificare le impostazioni dell'applicazione;
- *util*: contiene le classi di utilità generale, come ad esempio l'adapter per la connessione al database, oppure metodi per la creazione di AlertDialog. Al suo interno sono presenti quattro classi:
  - *Util*: contiene metodi utili per la creazione di AlertDialog semplici oppure personalizzate;

- *GestioneSocket*: gestisce la connessione al socket, la chiusura e vari parametri utili per la connessione;
- *DatabaseHelper*: gestisce la creazione e l'aggiornamento della struttura del database;
- *ConnessioneRecenteDatabaseAdapter*: gestisce l'inserimento, la ricerca, la modifica e l'eliminazione delle connessioni recenti all'interno del database.

➢ **GestioneSocket è una classe che contiene solamente metodi statici per la gestione del socket. Questo ci permette di avere un riferimento al socket sia nella MainActivity (per la creazione/chiusura del socket), sia in HotPotatoes (per prelevare gli stream di lettura e scrittura) dato che non era possibile passare il socket tramite l'Intent perché la classe Socket non implementa l'interfaccia Serializable.**

Le strutture di salvataggio dei dati usate dall'applicazione sono due:

- Un database SQLite che permette il salvataggio delle connessioni recenti. Il database contiene un'unica tabella: Connessione_Recente(**_id,** nome, indirizzo_ip, porta, timestamp). Il campo timestamp è utilizzato per ordinare le connessioni recenti in ordine cronologico di utilizzo e viene aggiornato ogni volta che una connessione recente viene effettuata;
- Le SharedPreference che sono un costrutto fornito da Android per il salvataggio di informazioni semplici (come ad es. valori booleani, stringhe, interi, ecc…) e visibili facilmente da tutte le classi dell'applicazione. Nell'app in esame vengono utilizzate per il salvataggio delle opzioni di visualizzazione dei dati ricevuti dal robot.

Tra i vari sensori di cui l'Acer ICONIA A500 è dotato, si è scelto di utilizzare il sensore *orientation*. Questo sensore fornisce informazioni riguardo all'inclinazione laterale (ritornando valori compresi tra +180 e -180) e al beccheggio (inclinazione avanti/indietro, ritornando valori compresi tra +90 e -90) del dispositivo, oltre a fornire l'angolo dell'attuale posizione rispetto al nord magnetico (ritornando valori compresi tra 0 e 360 ma inutilizzato in questo elaborato). Il sensore ricava le sue informazioni utilizzando in maniera combinata altri due sensori: il sensore geomagnetico (che calcola i valori del campo magnetico terrestre nella posizione attuale) e l'accelerometro, che misura la forza applicata al dispositivo (compresa la forza di gravità). Rispetto ad altri sensori (quali ad esempio l'accelerometro) *l'orientation* presenta il grosso vantaggio di fornire valori già ripuliti dal rumore elettrico, generato dai circuiti che implementano i sensori utilizzati.

➢ **Per trasformare i valori ritornati dal sensore si è deciso di limitare l'area di lavoro, passando da 360° a 60° sia per la rotazione laterale sia per il beccheggio (cioè considerando massima una rotazione di 30° in avanti, indietro, a destra o a sinistra rispetto alla posizione iniziale), in modo da facilitare l'accelerazione del robot. Successivamente si esegue una trasformazione lineare che limita i valori ritornati dal sensore tra -1 e 1. Infine, questi dati vengono elevati al cubo in modo da aumentare il numero di valori del sensore che corrispondono a basse velocità (per una partenza più dolce) e diminuire invece quelli relativi ad alte velocità.**

3.2.1. Activity Principale

Quest'activity permette l'inserimento di tutti i parametri necessari alla connessione ad un server e gestisce anche il collegamento, notificando all'utente tutti gli errori riscontrati durante la connessione. Risulta essere composta da 3 parti distinte.

La prima parte (riquadro blu nella figura) richiede all'utente tutti i parametri necessari per la connessione (indirizzo ip, porta e un parametro opzionale che è il nome della connessione, utilizzato solamente per permettere all'utente una rapida ricerca nelle connessioni recenti) e notifica gli errori presenti nei dati inseriti.

La seconda parte (riquadro giallo) contiene una lista di connessioni recenti, cioè connessioni correttamente eseguite, ordinate in ordine cronologico di utilizzo. In questa parte è possibile premere su una qualunque delle connessioni recenti per avviare in modo automatico la connessione al server.

La terza e ultima parte (riquadro rosso) permette di eseguire l'activity che modifica le impostazioni dell'applicazione.

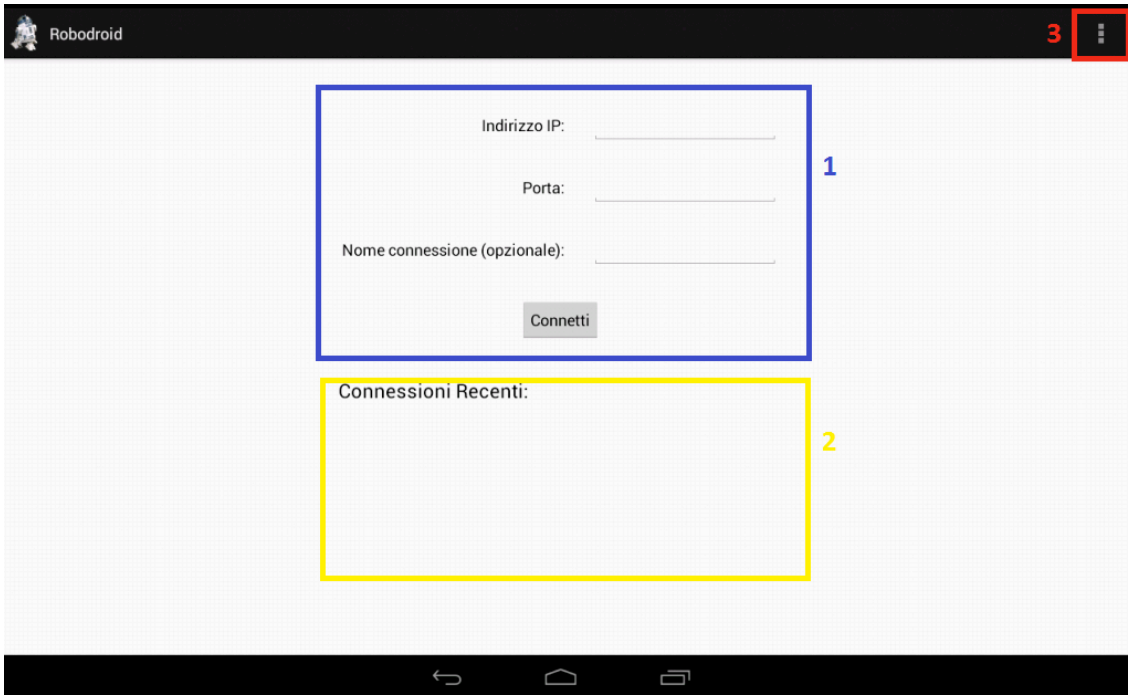

Figura 3: Activity principale

3.2.2. Activity di controllo

Quest'activity viene richiamata quando si è stabilita correttamente la connessione al server. In sostanza permette l'invio dei comandi di movimento al server e la visualizzazione dei dati ricevuti dal server. Risulta essere formata da 6 parti distinte.

La prima parte (riquadro arancio in figura) permette l'abilitazione o meno del controllo delle collisioni del robot.

La seconda parte (riquadro giallo) permette di visualizzare tutti i dati ricevuti dal robot (come ad esempio indirizzo IP, velocità effettiva del robot, ecc…). I dati mostrati sono quelli che l'utente ha deciso di visualizzare nelle impostazioni dell'applicazione. Per nascondere un determinato campo, basta entrare nelle impostazioni e deselezionare la checkbox relativa alla visualizzazione del campo che si vuole nascondere.

La terza parte (riquadro azzurro) permette di fermare il controllo del robot. Se si preme il bottone "STOP!", il robot si ferma ma il collegamento al server rimane attivo e viene presentata all'utente una finestra di dialogo in cui è possibile scegliere se riprendere il controllo oppure chiudere la connessione. È importante notare che la ripresa del controllo del robot non comporta una ricalibrazione automatica del sensore.

La quarta parte (riquadro rosa) permette di ricalibrare il sensore in modo da definire una nuova posizione iniziale del tablet, più comoda rispetto alla precedente. La ricalibrazione viene effettuata presentando all'utente l'Alert Dialog descritto nella sesta parte di quest'activity.

La quinta parte (riquadro verde) permette di capire l'orientamento del tablet rispetto alla sua posizione iniziale o alla ricalibrazione effettuata dall'utente. È formata da una view personalizzata che disegna una pallina verde su un radar circolare posizionata seguendo l'inclinazione del tablet.

La sesta parte (riquadro rosso) serve per permettere una ricalibrazione del sensore e viene presentata sia quando l'utente richiede una ricalibrazione del tablet (usando il bottone presente nella

parte quattro), sia al primo avvio di questa activity permettendo una calibrazione iniziale prima di comandare veramente il robot.

Sono presenti altre parti che non vengono utilizzate a causa di problemi riscontrati durante le varie prove di streaming video ma che comunque potranno essere riciclate per altri scopi.

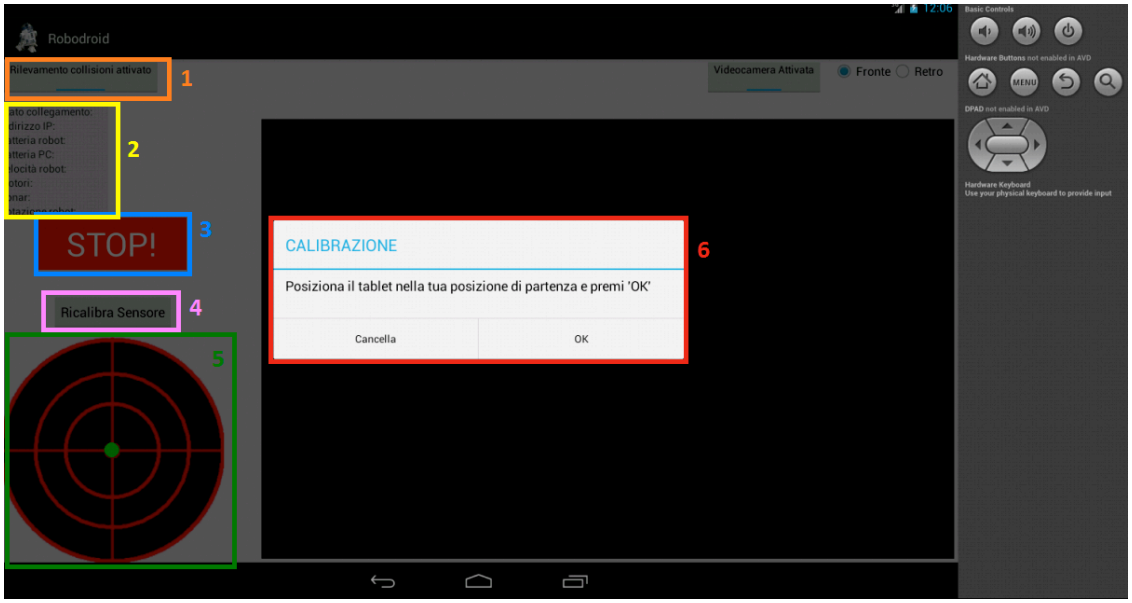

Figura 4: Activity di controllo

3.2.3. Activity delle impostazioni

Quest'activity permette di gestire le impostazioni dell'applicazione. In sostanza è possibile eliminare la lista delle connessioni recenti oppure nascondere alcuni dati ricevuti dal server. È composta da due parti.

La prima parte (riquadro rosso nella figura) permette di visualizzare i dettagli della categoria selezionata nella seconda parte. Nel caso dell'applicazione in esame è possibile visualizzare o una serie di checkbox per decidere gli elementi da visualizzare opure un bottone che permette l'eliminazione di tutte le connessioni recenti.

La seconda parte (riquadro blu) permette di selezionare una categoria i cui dettagli verranno espansi nella prima parte.

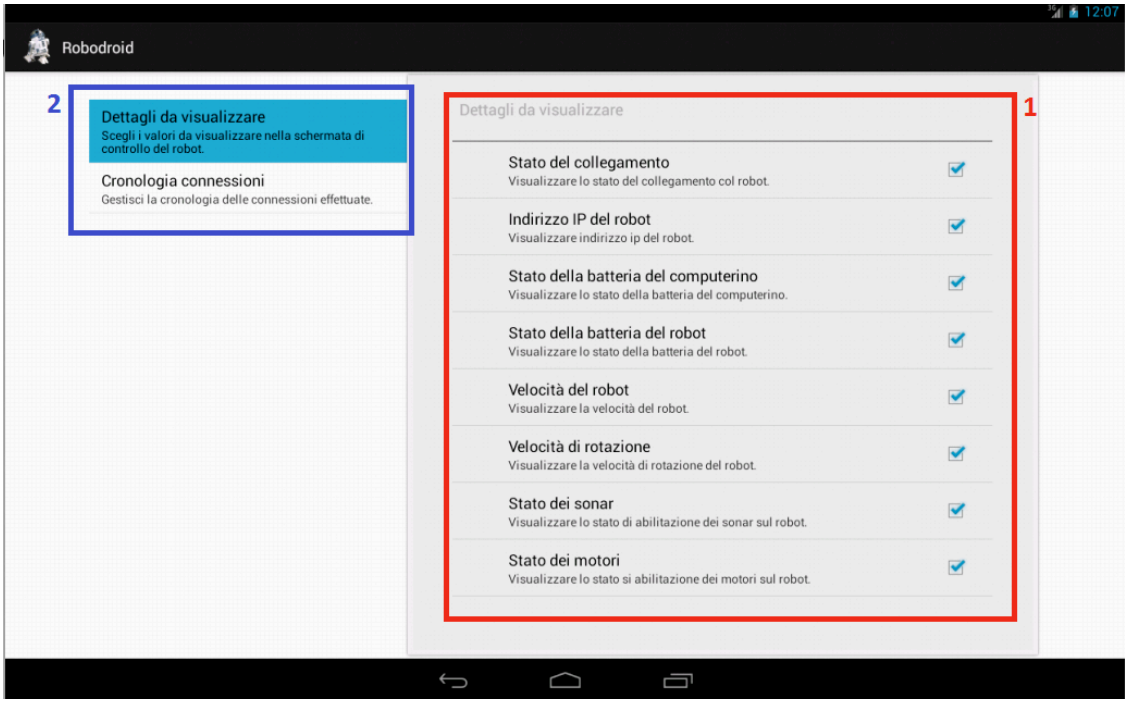

Figura 5: Activity delle impostazioni

3.3. Comunicazione

La comunicazione tra client e server avviene tramite il protocollo TCP e tramite socket. In sostanza, il client invia al server (ogni 300ms) un insieme di caratteri nella seguente forma “0.12 0.34 1” dove:

- Il primo è un numero compreso tra -1 e 1, con due cifre decimali e rappresenta la velocità lineare da impartire al robot. Se è positivo il robot si muoverà in avanti, se invece è negato si muoverà all’indietro. Il server moltiplicherà questo numero con la velocità lineare massima del robot (ricavata attraverso le librerie Aria) in modo da rendere il protocollo robot-indipendente;
- Il secondo è un numero compreso tra -1 e 1, con due cifre decimali e rappresenta la velocità di rotazione da impartire al robot. Se è positivo il robot ruoterà in senso orario rispetto al suo asse, se invece è negativo si muoverà in senso antiorario. Il server moltiplicherà questo numero con la velocità di rotazione massima del robot (ricavata attraverso le librerie Aria) in modo da rendere il protocollo robot-indipendente;
- Il terzo è un numero che può valere 0 o 1 e rappresenta a volontà di abilitare o meno il controllo delle collisioni del robot.

Il server, invece, invia al client un insieme di caratteri nella seguente forma: “12.34 1 0 123.45 678.90 123” dove:

- Il primo è un numero con la virgola che rappresenta i Volt di carica della batteria del robot;
- Il secondo è un numero intero che può valere 0 o 1 e indica lo stato di abilitazione dei motori;
- Il terzo è un numero intero che può valere 0 o 1 e indica lo stato di abilitazione dei sonar;
- Il quarto è un numero con la virgola che indica la velocità di rotazione effettiva del robot (espressa in mm/s);
- Il quinto è un numero con la virgola che indica la velocità lineare effettiva del robot (espressa in mm/s);
- Il sesto è un numero intero che indica la percentuale di batteria del calcolatore attaccato al robot.

- **Sia il server sia il client considerano "caduta" la connessione nel caso in cui vengono persi 5 pacchetti consecutivamente. Un pacchetto si considera "perso" se la read sul socket non riceve nulla in 750ms oppure se il pacchetto subisce un'alterazione (ad es. se la sua lunghezza o la sua forma non sono conformi con quella stabilita nel protocollo descritto poche righe sopra).**

Come protezione aggiuntiva, si è pensato di azzerare le velocità del robot nel caso in cui il server consideri "perso" un pacchetto.

4. Modalità operative

4.1. Componenti necessari

Per il corretto funzionamento del sistema sono necessari un tablet con Android 3.2 o successivo su cui deve essere installata l'applicazione Robodroid e un server scritto in linguaggio C++ che deve essere installato sul pc di controllo del robot.

L'applicazione Android è stata sviluppata usando l'IDE Eclipse e l'Android SDK. E' possibile scaricare dal sito http://developer.android.com/sdk/index.html una pacchetto che contiene entrambi gli strumenti, oppure se si dispone di una copia di Eclipse già installata sul calcolatore è possibile aggiungere il repository dell'Android SDK alle sorgenti di aggiornamento software del programma e in eseguito scaricare il pacchetto direttamente all'interno di Eclipse.

Il server in esecuzione sul robot è stato creato utilizzando il linguaggio C++. Per compilarlo sono sufficienti i normali strumenti di sviluppo C++ presenti nelle distribuzioni linux.

4.2. Modalità di installazione

L'applicazione Android viene compilata da Eclipse in un pacchetto con estensione **.apk**, che occasionalmente potrebbe essere associato ad alcuni programmi di compressione installati sul pc. Per installare l'applicazione si deve copiare il pacchetto sul tablet, raggiungerlo con un file manager ed eseguirlo. Prima è necessario tuttavia abilitare l'installazione di applicazioni provenienti da sorgenti sconosciute. Generalmente questa opzione, che si presenta come una checkbox normalmente disattivata, la si può trovare nel sottomenu **"Sicurezza"** dentro alle impostazioni del tablet.

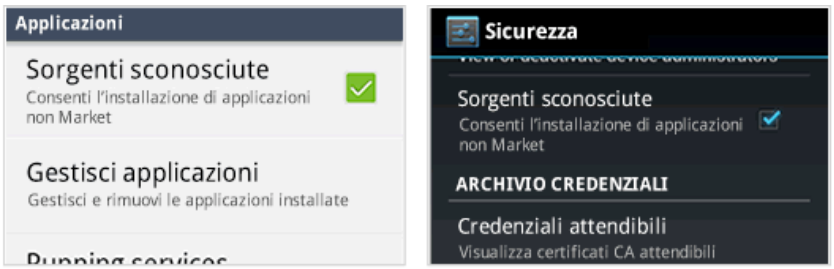

Figura 6: Screenshot dell'abilitazione delle sorgenti sconosciute

Esiste un altro modo per installare l'applicazione sul tablet. All'interno di Eclipse, quando si avvia l'esecuzione del progetto, l'IDE chiede su quale dispositivo si vuole eseguire l'applicazione. A questo punto è sufficiente selezionare il tablet reale invece dell'emulatore e l'applicazione viene scaricata sul dispositivo e installata automaticamente. Questo naturalmente è possibile solo se si è in possesso del codice sorgente. Se ci si trova nell'ambito del laboratorio è ovviamente meglio usare la seconda modalità, invece se si vuole distribuire l'applicazione già compilata a terzi è necessario usare la prima modalità.

Il server C++ può essere inviato al calcolatore del robot tramite i classici strumenti di Linux, come il comando "**scp"** e ci si può connettere al server usando il comando "**ssh"**. Per compilare il codice sorgente bisogna posizionarsi nella directory del server ed eseguire in successioni i comandi

make

per compilare il server e

make install

per installarlo in modo che sia eseguibile da qualsiasi percorso. Per lanciare il server si deve digitare ed eseguire il seguente comando:

- **serverAndroid -rp /dev/ttyUSB[0-9]**

4.3. Modalità di taratura

La posizione centrale del sistema di guida viene impostata al momento della connessione come la posizione attuale in cui si trova il tablet. È quindi opportuno che l'utilizzatore dell'applicazione, prima di procedere alla connessione con il robot, posizioni il tablet nelle sue mani nella posizione più comoda e abituale. In ogni momento è comunque possibile ritarare il sistema di guida premendo il pulsante "**Ricalibra sensore"** che chiede all'utente di posizionare il tablet nella posizione di partenza.

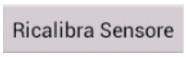

Figura 7 Bottone ricalibra sensore dell'applicazione

Una volta posizionato il tablet, confermare la scelta e l'applicazione leggerà i dati dell'accelerometro e li userà come posizione inziale per tutti i futuri spostamenti fino a una nuova connessione o a una nuova ricalibratura.

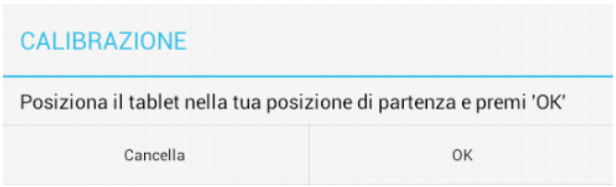

Figura 8: Alert Dialog per la conferma della ricalibrazione

4.4. Avvertenze

Le prime volte che si usa l'applicazione è necessario prendere confidenza con il robot e in particolare effettuare movimenti dolci e vicini all'origine del sistema di guida. I robot del laboratorio sono comunque dotati di strutture atte ad attutire eventuali collisioni ma naturalmente è sempre meglio impedire una tale eventualità. Se ci si rende conto di non riuscire a riprendere il controllo del robot per evitare una collisione è possibile effettuare uno stop di emergenza premendo sul pulsante "**STOP!"**.

Figura 9: Bottone stop dell'applicazione

Come già detto in precedenza, il robot si ferma immediatamente ma la connessione rimane attiva, e viene data la possibilità di scegliere se riprendere il controllo oppure arrestare la connessione.

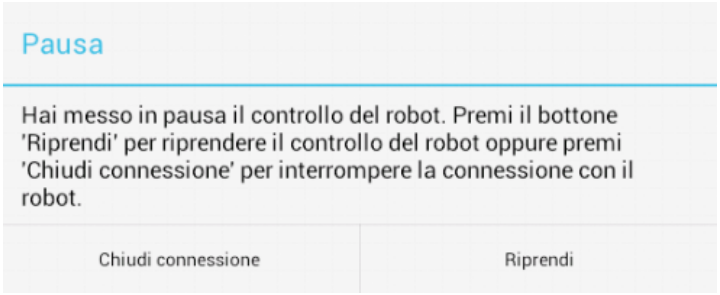

Figura 10: Alert Dialog presentata dopo la pressione del tasto Stop!

E' inoltre fortemente consigliato di lasciare attivo il sistema automatico di rilevamento delle collisioni, in questo modo il server può rallentare e fermare, nella maggior parte dei casi, in tempo il robot se si accorge di un ostacolo usando il sonar.

Per il collegamento al server, bisogna conoscere l'indirizzo ip del calcolatore al quale è attaccato il robot. Questo indirizzo ip viene acquisito dinamicamente e quindi è necessario controllare ad ogni connessione che sia corretto.

5. Conclusioni e sviluppi futuri

Il progetto ci ha permesso di apprendere lo sviluppo delle applicazioni per il sistema operativo Android e come usare un tablet per comandare un robot. Questo apre scenari per futuri miglioramenti, ad esempio si potrebbe telecomandare un robot (magari addetto alla sorveglianza) direttamente dal proprio telefono cellulare o tablet. Sarebbe interessante appunto testare l'applicazione su un telefono cellulare. Il protocollo di comunicazione può essere espanso con nuovi comandi e costruendo l'opportuna interfaccia utente sull'applicazione e modificando il codice del server si possono supportare nuovi robot oppure nuovi comportamenti per il robot di test.

Un altro possibile sviluppo futuro, sempre nel caso della sorveglianza remota, sarebbe quello di lasciare l'applicazione sempre attiva in background e fare in modo che possa riattivarsi qualora il robot rilevasse qualcosa di sospetto.

Infine, è possibile utilizzare la videoview presente nell'activity di controllo in modo da catturare lo streaming video delle videocamere del robot, oppure visualizzare un radar che permette di mostrare una rappresentazione grafica dei valori ritornati dai vari sonar o dal sensore laser.

Bibliografia

[1] Corso di Android a cura di Carlo Pelliccia (15 lezioni): http://www.informatica.uniroma2.it/upload/2009/LIS/

[2] Pagina documentazione librerie Android: http://developer.android.com/reference/packages.html

[3] Sistema operativo Debian: http://www.debian.org

[4] Descrizione Android: http://developer.android.com/guide/components/index.html

[5] Mailing list Debian: http://lists.debian.org

[6] Funzionamento Debian: https://en.wikipedia.org/wiki/Operative_system

[7] Funzionamento Android: https://en.wikipedia.org/wiki/Android_%28operating_system%29

[8] Forum generale di supporto: http://stackoverflow.com

Indice