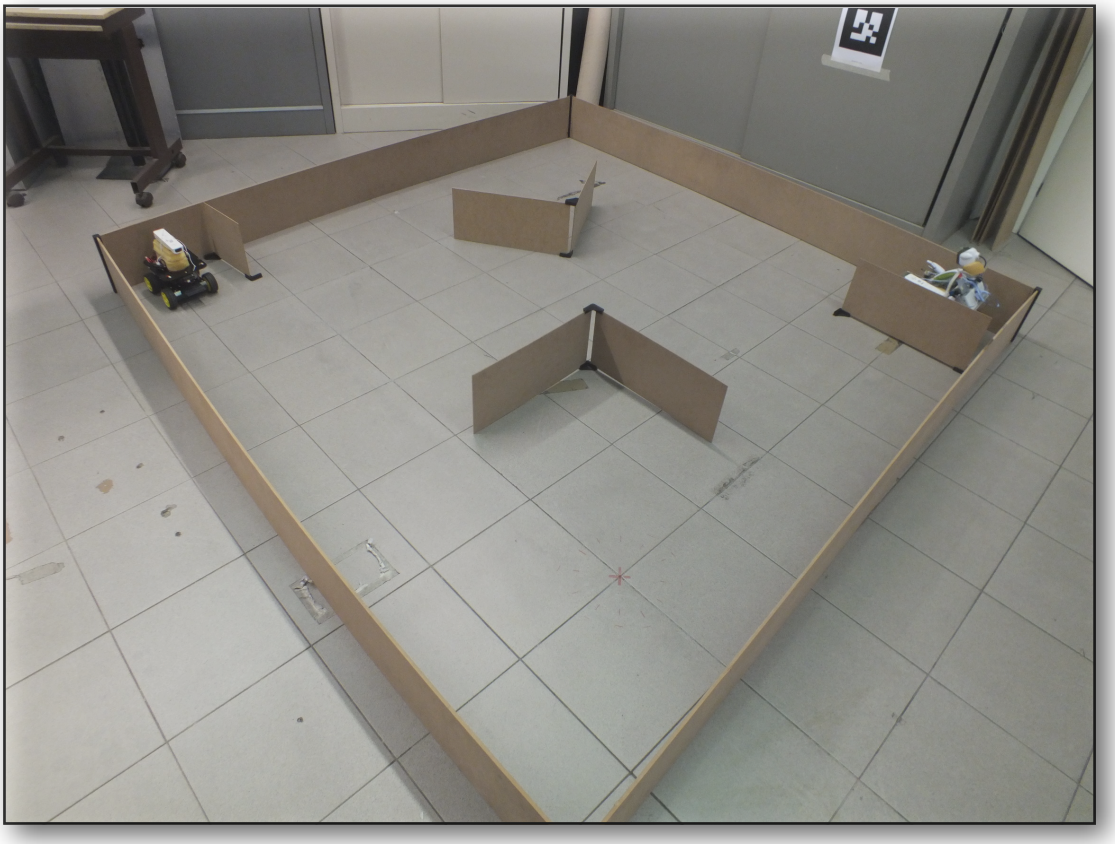
Laser Game Bot

Progetto di Ingegneria Informatica (5 CFU)
Robogames

Responsabile: Andrea Bonarini
Anno Accademico: 2013-2014



Fotografia del labirinto di gioco con i robot pronti a cominciare
(a sinistra il robot autonomo, a destra il robot guidato dal giocatore)

Prefazione

La descrizione del progetto è a parte, presente nella pagina *Discussion* del progetto.
Di seguito vengono riportate le descrizioni dei componenti, degli strumenti e del materiale utilizzato per la preparazione e l'attuazione del gioco.

Componenti

Robot Autonomo

Il robot autonomo funge da avversario nel gioco; si muove casualmente nel percorso seguendo e distanze che riceve dai sensori, con lo scopo di trovare il robot del giocatore e colpirlo.

Hardware

Il robot di base è un ArduQuad, su cui sono montati diversi componenti:

- Un Arduino Uno [che gestisce la logica di movimento, l'accensione dei LED, i sensori di distanza, la trasmissione e la ricezione di dati dal computer.
- Una scheda di movimento TB6612FNG che controlla i 4 motori mediante un canale potenza e due di direzione.
- Un XBee Pro che permette la comunicazione Bluetooth con il computer, per ricevere i segnali corrispondenti ai colpi subiti dal robot.
- Un WIIMote connesso al computer, mediante il quale si può colpire il giocatore per eliminarlo dal gioco.
- Sensori di distanza Sharp GP2D120XJ00F con cui vengono prese le distanze dai muri del percorso.
- LED infrarossi, che fungono da bersaglio per il wiimote.

Il robot inoltre, per funzionare correttamente, ha bisogno di piena potenza della batterie (7,5V) sia per i motori che per gli altri componenti.

Software

La logica del robot è scritta in codice di Arduino, mediante l'utilizzo della sua IDE (scaricabile dal sito arduino per il proprio conmputer).
Le diverse schede contengono le diverse funzioni svolte con le proprie librerie.

- *mainRobogame* contiene l'inizializzazione e il ciclo di gioco.
- *LED* gestisce le funzioni per accensione e spegnimento dei led.
- *MotorControl* gestisce i movimenti del robot, utilizzando la scheda
- *Sensors* controlla i sensori di distanza, mediante un libreria trovata in rete di un progetto precedente (RoverCommander, presente nel sorgente).
- *Roaming* controlla i movimenti del robot, utilizzando i sensori ed una libreria trovata in rete ed adattata alla scheda motori.
- *provaXbee* gestisce la comunicazione mediante Xbee.

Ci sono inoltre altre due librerie (provaRoamingWithSensors, proveFinali) che contengono codice usato nelle varie prove effettuate.

Note

Solo nel momento di pieno funzionamento del robot si è capito che bisogna inserire dei controlli sulle distanze minime a destra e sinistra in più durante i movimenti.

Robot Controllato dall'utente

Ul robot controllato dall'utente ha il ruolo di simulare la presenza del giocatore all'interno del labirinto, con lo scopo di cercare l'avversario, colpirlo senza essere colpiti a propria volta, nascondersi per permettere alla propria arma di ricaricarsi.

Hardware

Il robot utilizzato è uno Spykee, a cui sono stati aggiunti alcuni componeti:

- Un WIIMote connesso al computer, mediante il quale si può colpire l'avversario per vincere il gioco.
- LED infrarossi, che fungono da bersaglio per il wiimote.

Software

Il robot viene controllato con il proprio software proprietario (scaricabile dal sito e presente nel codice sorgente), installato su di un secondo computer, mediante cui il giocatore vede il labirinto attraverso la telecamera presente sulla testa del robot.
{ Questa soluzione è stata adottata per necessità, in quanto il codice è eseguibile solamente su Windows/Mac, non funzionante né su macchina virtuale né su emulatori come PlayOnLinux o Wine e non c'è stato tempo per modificare tutta l'architettura del progetto rendendolo eseguibile su Linux.}

I LED invece, essendo stati inseriti a parte sul robot, sono collegati alla scatola presente sulla schiena del robot e devono essere accesi in modo separato; bisogna collegare il dispositivo Xbee sincronizzato con quello presente sul robot al computer, aprire un canale di comunicazione mediante la porta USB specifica (si può utilizzare un programma di comunicazione terminale come Cutecom, aprendolo sulla porta corretta a frequenza 115200) , accenderli mediante il comando

```
infrared on
```

e controllare che il comando sia andato a buon fine (si può usare il codice eseguibile *provaWiiEx*; sincronizzando il wiimote, attivando il tracciamento infrarossi e verificare) altrimenti riprovare o riaccendere la scatola.

Codice eseguito su PC

Il codice è stato scritto in linguaggio C, sistema operativo Linux e sfrutta alcune librerie aggiuntive:

- Wiiuse.h per la gestione dei wiimote attraverso comunicazione Bluetooth.
  [http://www.macs.hw.ac.uk/~ruth/year4VEs/Labs/wiiuse.html]
  [https://github.com/rpavlik/wiiuse]
- NCurses.h per la gestione della pressione dei tasti senza invio per simulare lo sparo.

Entrambe le librerie devono essere installate sul computer previa compilazione.
Viene poi eseguito da terminale mediante compilatore gcc aggiungendo i link alle librerie

```
gcc nomeCodice.c -o nomeEseguibile -lwiiuse -lncurses
```

Inoltre viene sfruttato un adattatore USB/Seriale che deve essere inserito nella porta `/dev/ttyUSB0` del computer (nel caso bisogni utilizzare un'altra porta bisogna modificarla nel codice di comunicazione seriale, inserendo quella corretta).
Eseguire poi mediante il comando

```
./nomeEseguibile
```

Il gioco è presente in due versioni: una con una funzione di sparo da parte del giocatore, rendendo il gioco più interattivo ma molto più difficile (riuscire a centrare i LED non è semplicissimo) ed una in cui lo sparo invece è automatico, più semplice ma comunque divertente.

Note

Solo nel momento della valutazione e di pieno funzionamento del robot si è capito che la logica di controllo degli eventi wiimote non è pienamente soddisfacente; mettendo i processo in *sleep* (ricarica dell'arma) si crea una coda di eventi che vengono controllati singolarmente, accumulando ritardi su ritardi e non permettendo di capire se il colpo è andato a segno subito o meno.

Possibili sviluppi

- Migliorare la logica di controllo degli spari nel codice C, per renderlo più reale ed evitare una coda di eventi controllati singolarmente.
- Migliorare la logica di movimento in Arduino, inserendo dei controlli sulle distanze minime da destra e sinistra (tolti per malfunzionamenti precedenti).
- Integrazione del sistema di controllo di Spykee con il resto del software, utilizzando ROS per creare i nodi di controllo e modificando il codice di *RoboTower* per i movimenti del robot.