Politecnico di Milano
Scuola di Ingegneria Industriale e dell'Informazione
Corso di Laurea Magistrale in Ingegneria Informatica
Dipartimento di di Elettronica, Informazione e Bioingegneria

KartBot: Sviluppo di un Racing Game con Robot Autonomi

AI & R Lab
Laboratorio di Intelligenza Artificiale
e Robotica del Politecnico di Milano

Relatore: Prof. Andrea BONARINI

Tesi di laurea di:
Elisa DUI Matr. 782949
Davide TESSARO Matr. 783851

Anno Accademico 2012–2013

Alle nostre famiglie

Ringraziamenti

Entrambi vorremmo ringraziare innanzitutto il nostro relatore, il professor Andrea Bonarini, per la possibilità di lavorare in questo appassionante ambito, per la grande disponibilità, il costante aiuto ed i preziosi consigli.

Un enorme grazie va anche a tutti i ragazzi dell'AIRLab per la loro simpatia ed il sostegno nei momenti di difficoltà.

Sommario

La robotica occupa un posto di rilevanza tra i settori tecnologicamente avanzati in maggiore espansione negli ultimi anni. Dei molteplici ambiti di cui essa si occupa, uno è rappresentato dallo studio dei robogame, giochi realizzati mediante l'utilizzo di robot autonomi. Essi costituiscono l'evoluzione naturale dei videogame, considerando che la direzione intrapresa nel mondo ludico è finalizzata al costante miglioramento dell'interazione tra l'uomo e il gioco stesso, ottenuta attraverso la sua concretizzazione nel mondo reale.

Partendo dalle varie robot racing esistenti e seguendo la scia di questa trasformazione, abbiamo preso spunto da un famoso videogame, Mario Kart Wii, per realizzare *KartBot*, un robogame consistente in una gara tra due kart, uno controllato dal giocatore e l'altro dall'intelligenza artificiale, resa più movimentata dalla possibilità di utilizzare armi inusuali per ostacolare l'avversario.

L'aspetto innovativo è dato dalla capacità del kart autonomo di navigare in un ambiente fortemente dinamico svolgendo l'intera elaborazione a bordo in tempo reale e utilizzando unicamente informazioni provenienti dai sensori e dal sistema di visione.

Il risultato ottenuto è stato la creazione di un gioco divertente, appassionante ed intuitivo, in grado di suscitare l'interesse di tutti, mediante una fusione tra tradizione e innovazione.

Abstract

Robotics is one of the most rapidly growing sectors in tecnological area. One of its topics is Robogame, which aims to create interactive games between humans and robots, exploting their autonomy and artificial intelligence algorithms to obtain challenging and interesting situations, using cheap technologies. Robogames can be thought as the natural evolution of videogames, keeping in mind that the growing line followed by the most famous videogame companies is pointed to let the player feeling physically participating in the game.

Taking care of these aspects, we created *KartBot,* a robogame consisting in a race between two karts, one controlled by a player and the other one by artificial intelligence, where both of them can get strange weapons to obstacle their opponent trying to win the race.

The most innovative aspect in *KartBot* consists in the autonomous kart, which is able to move by itself in a dynamic environment, exploiting only sensor perceptions, images and computer vision algorithms, processed by an on board mini computer.

KartBot is an amazing, exciting and easy to use game, able to keep in interest everyone, by a well-balanced combination of tradition and innovation.

Indice

Elenco delle figure

Elenco delle tabelle

Capitolo 1

Introduzione

Viviamo in un mondo in cui la tecnologia pervade ormai ogni aspetto delle attività umane. Dalle industrie alle case, dagli uffici al tempo libero, la presenza di oggetti più o meno tecnologici è diventata una componente fondamentale della nostra esistenza. In questo contesto si inserisce la robotica, seppur le sue applicazioni commerciali siano ancora agli albori rispetto all'incredibile avanzamento di altri settori. Gli ambiti di cui essa si occupa sono molteplici: chirurgia, industria, esplorazione di luoghi sconosciuti, zone militari o territori colpiti da catastrofi, intrattenimento, riabilitazione, protesi, veicoli autonomi e molti altri. Noi abbiamo deciso di concentrarci su quello dei giochi interattivi basati sull'utilizzo di robot: i *robogame*.

I robogame possono essere visti come la nuova frontiera dei videogame, un modo innovativo di concepire il gioco, così da permettere al giocatore di interagire fisicamente con esso. L'idea di videogame aveva già subito una prima trasformazione a seguito dell'introduzione di dispositivi, come Wii Remote, Kinect, Move e affini, i quali hanno portato allo sconvolgimento del paradigma di interazione uomo-macchina: al centro dell'esperienza di gioco non vi è più quello che succede al suo interno, ma piuttosto quello che il giocatore fa. Oggi si assiste ad un'ulteriore evoluzione: il gioco tecnologico non è più relegato dietro allo schermo di un computer o di un televisore, ma prende forma e si concretizza nel mondo reale. Lo scopo primario di un robogame diventa, quindi, quello di catturare l'attenzione del giocatore, suscitandone l'interesse e coinvolgendolo, a livello fisico, mentale ed emotivo. Il mezzo per realizzare giochi con una così elevata capacità di interazione è quello di utilizzare robot autonomi, possibilmente costruiti con tecnologie poco costose, in modo da

poter idealmente entrare in casa di ognuno.

Un robogame può avere, come soggetto, un qualunque tipo di gioco. L'ambito che ci ha maggiormente interessato e nel quale abbiamo deciso di lavorare è quello delle robot racing, gare tra robot autonomi con sembianze di automobili, che si sfidano a percorrere tracciati di complessità differente. Lo scopo della tesi è stato, quindi, quello di realizzare un gioco, che abbiamo deciso di chiamare *KartBot*, in grado di unire questo genere di competizioni a una classica corsa con le macchinine telecomandate e, per fare ciò, ci siamo ispirati ad un videogame molto noto: Mario Kart Wii™.

La particolarità di KartBot è rappresentata dal fatto che l'elaborazione necessaria alla navigazione del kart autonomo avviene completamente a bordo di esso in tempo reale, basandosi esclusivamente sulle informazioni ricevute mediante i sensori ed il sistema di visione.

KartBot consiste in una gara su pista tra due *kart,* caratterizzata dalla possibilità di ottenere e utilizzare armi inusuali, al fine di ostacolare l'avversario e ottenere la vittoria. I kart sono rappresentati da robot su ruote, uno comandato tramite il controller Wii Remote e l'altro autonomo. La competizione si svolge su una pista appositamente disegnata e comincia allo scattare del semaforo. Il gioco è corredato da un'applicazione, tramite la quale vengono visualizzate alcune informazioni importanti: un video con ciò che viene visto dal kart durante la gara, la posizione occupata costantemente aggiornata e le armi a disposizione del giocatore, ottenute passando con il kart sopra i relativi indicatori presenti sulla pista. La gara termina quando il primo giocatore raggiunge il traguardo.

In tutto lo sviluppo del gioco abbiamo fatto in modo che le decisioni prese permettessero di mantenere la massima flessibilità ed espandibilità. Idealmente, infatti, il gioco potrebbe essere ampliato, aumentando il numero di kart, sia autonomi sia controllati dai giocatori, oppure incrementando, ad esempio, la tipologia delle armi presenti. Anche il tracciato della pista può essere ridisegnato in forme più elaborate, non essendo questo vincolante per il funzionamento del kart autonomo.

L'obiettivo che ci siamo posti e che ha guidato ogni nostra scelta implementativa è stato quello di realizzare un gioco il più possibile coinvolgente e movimentato. Effettuando una fusione tra tradizione e innovazione, abbiamo concepito un gioco fatto di regole semplici, ma con quel pizzico di originalità in grado di suscitare l'interesse di tutti coloro, dai bambini agli adulti, che ancora

amano giocare.

1.1 Struttura della tesi

La tesi è strutturata nel modo seguente:

Capitolo 2 - Stato dell'arte: Si presentano le origini dei robogame e lo stato attuale del loro sviluppo e si analizzano i progetti simili a KartBot.

Capitolo 3 - Descrizione del gioco: Si descrivono il gioco e il regolamento in dettaglio, si elencano le funzionalità del kart, si riporta il progetto dell'ambiente di gioco.

Capitolo 4 - Architettura hardware: Si descrivono i moduli hardware che compongono il kart.

Capitolo 5 - Architettura software: Si descrivono i moduli software che compongono il kart.

Capitolo 6 - L'applicazione del gioco: Si illustra l'applicazione di gioco e l'interazione di essa con i kart.

Capitolo 7 - Test e risultati: Si elencano i test svolti e i risultati raggiunti.

Capitolo 8 - Conclusioni: Si riportano le conclusioni e i possibili sviluppi futuri del gioco.

Appendice A - Configurazione delle unità di elaborazione: Si illustrano le unità di elaborazione usate e la loro configurazione.

Appendice B - Istruzioni di montaggio: Si riportano gli schemi circuitali e le istruzioni per il montaggio.

Appendice C - Documentazione del software: Si presentano i diagrammi UML e la documentazione del software.

Appendice D - Manuale d'uso: Si riporta il manuale d'uso per l'utente finale.

Capitolo 2

Stato dell'arte

2.1 Evoluzione del gioco: dai videogame ai robogame

Alzi la mano chi non ha mai giocato ad un videogame. Bambini, adulti, sfide tra amici o con perfetti sconosciuti: è difficile trovare qualcuno che sia rimasto estraneo a questo mondo. I videogame rappresentano il più travolgente fenomeno di massa degli ultimi vent'anni, un potente mezzo di attrazione, in grado di rapire la mente dei giocatori, immergendoli in un universo virtuale. Il sociologo Alberto Abruzzese [1] ha affermato che essi sono "la nostra più avanzata frontiera e il nostro più affascinante futuro" [2]. I videogame possiedono, infatti, un elevato potenziale, essendo strettamente legati allo sviluppo tecnologico e, come tali, sottoposti a continue evoluzioni. Ciò ha portato le grandi aziende del settore a ricercare dispositivi sempre più innovativi, in grado di modificare il paradigma originario di interazione tra l'uomo e la macchina, per catturare il giocatore in modo più coinvolgente e rendere la sua risposta il più naturale e intuitiva possibile.

Tutto ha avuto inizio con la trasformazione delle interfacce di gioco: dai tradizionali joypad si è passati a controller che replicassero in modo più o meno fedele l'aspetto di oggetti reali. Il primo di questi ad aver ottenuto una vasta distribuzione commerciale è rappresentato dalla chitarra creata per il videogame Guitar Hero (*Figura 2.1*), pubblicato nel 2005 per PlayStation 2.

Lo sconvolgimento più grande, però, si è avuto nel 2006, con la presentazione da parte di Nintendo della console Wii, rivelatasi espressione di una visione rivoluzionaria. Essa era accompagnata da una nuova generazione di controller, il Wii Remote (*Figura 2.2a*), collegato ad essa mediante bluetooth e munito

Figura 2.1: Guitar Hero

(a) Wii Remote (b) PlayStation Move (c) XBox360 Kinect

Figura 2.2: Controller

di sensori in grado di tracciare parametri fisici, quali posizioni, accelerazioni e rotazioni, per rilevare i movimenti effettuati dal giocatore e riportarli all'interno dei giochi al fine di controllare personaggi o eventi. La risposta degli altri grandi colossi del settore è arrivata con qualche anno di ritardo. Nel 2010 Sony ha immesso nel mercato un controller molto simile a quello di Nintendo, il PlayStation Move (*Figura 2.2b*). Nello stesso anno Microsoft ha presentato Kinect per Xbox360 (*Figura 2.2c*), una webcam RGB, con doppio sensore di profondità a raggi infrarossi, in grado di rilevare i movimenti dell'intero corpo, la sua posizione e quella degli arti con precisione elevata e contenente anche un sistema per il rilevamento vocale. Successivamente sono stati introdotti anche altri dispositivi tecnologicamente più avanzati: l'evoluzione era cominciata.

Un concetto opposto al trasferire gesti e movimenti nella realtà virtuale è quello di portare il videogame fuori dalla cosiddetta quarta parete, permettendo al giocatore di vivere un'esperienza d'uso totalizzante, che lo renda concretamente parte di esso, mediante un feedback fisico sempre maggiore. Di questa categoria fanno parte le interfacce indossabili, che restituiscono perce-

(a) Google Glass (b) Oculus Rift

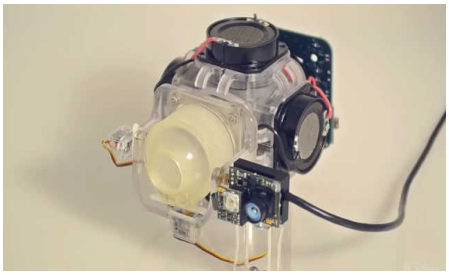
Figura 2.3: Visori HDM

zioni sensoriali di quello che sta avvenendo all'interno del gioco, come i guanti contenenti interfacce aptiche, che permettono di manipolare oggetti virtuali e ricostruiscono la sensazione data dal tocco, i corpetti, dotati di decine di sensori in grado di simulare colpi di armi da taglio e da fuoco, ed esoscheletri più completi, cioè tute piene di sensori in grado anche di rilevare qualunque movimento del corpo. I dispositivi indossabili che possiedono, però, le potenzialità maggiori per una vasta diffusione nel campo dei videogame sono i visori HDM (Head-Mounted Display): dai Google Glass (*Figura 2.3a*), che sfruttano la realtà aumentata, cioè la sovrapposizione di elementi virtuali al mondo reale, all'Oculus Rift (*Figura 2.3b*), che, grazie ad uno schermo ad alta definizione, crea un universo totalmente immaginario in cui immergersi. Esso è in grado di calcolare i più piccoli movimenti della testa, permettendo di spostare la visuale all'interno del mondo virtuale semplicemente ruotandola, di avvicinare lo sguardo a qualcosa per vederlo meglio, di guardare oltre un angolo o sporgersi verso il basso.

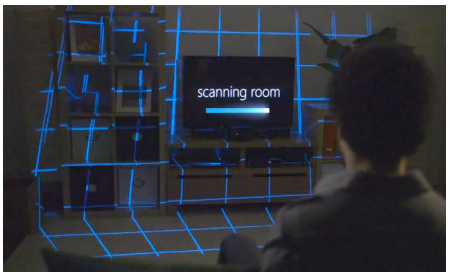
Esistono poi altri dispositivi particolari, pensati sempre con lo scopo di catturare maggiormente il giocatore, come Virtuix Omni (*Figura 2.4a*), un tapis roulant che, scorrendo a 360 gradi, permette di camminare all'interno del gioco usando le proprie gambe, o Disney Aireal (*Figura 2.4b*), che è in grado di generare mini-vortici d'aria, migliorando il feedback fisico ricevuto, o ancora IllumiRoom (*Figura 2.4c*), un'estensione del Kinect che trasforma l'intera stanza nel mondo all'interno del videogame.

Tutto ciò fa comprendere quanto sia vasto l'interesse nella ricerca di tecnologie sempre più innovative per migliorare l'esperienza di gioco, introducendo in alcuni casi radicali modifiche nella concezione del modo di giocare. La sfida

(a) Virtuix Omni



(b) Disney Aireal



(c) IllumiRoom

Figura 2.4: Nuovi dispositivi

è rendere i videogame sempre più concreti, permettendo al giocatore di diventarne parte e di interagire con essi nel mondo reale. Queste costituiscono alcune delle motivazioni che hanno portato alla nascita dei *robogame.*

I robogame sono ciò che meglio rappresenta il traguardo finale di tale evoluzione, in quanto permettono al giocatore di scendere in campo in prima persona, affrontando il nemico in una sfida concreta. Essi incarnano la necessità di riprendere il contatto con la realtà al di fuori dello schermo, eliminando limitazioni e barriere virtuali: non più solo spettatori, ma immersi in un gioco che è tutto intorno a sé. I robogame costituiscono un ritorno alle origini, ai giochi tradizionali, ma aggiungendo a questi qualcosa in più, in grado di stimolare maggiormente la curiosità e la fantasia. Perché un robot autonomo si trasformi in un vero compagno di giochi, quindi, occorre fondere aspetti appartenenti non solo alla robotica e all'intelligenza artificiale, ma anche a discipline quali la psicologia, le scienze cognitive, l'interazione uomo-macchina e l'estetica.

Lo scopo della ricerca nel campo dei robogame è realizzare giochi attrattivi e coinvolgenti, in grado di suscitare nei potenziali acquirenti il desiderio di possederli, utilizzando, però, tecnologie poco costose. Sono proprio i prezzi elevati, infatti, ad aver finora costituito il principale vincolo alla loro diffusio-

(a) Furby (b) Pleo (c) Aibo

Figura 2.5: Robogame commerciali

ne, limitata a pochi giochi commercializzati e diventati famosi, come Furby di Hasbro (*Figura 2.5a*), un esserino peloso in grado di apprendere, Pleo di Ugobe (*Figura 2.5b*), un cucciolo di dinosauro robot in grado di fare esperienza dell'ambiente che lo circonda e di sviluppare un'identità individuale e Aibo di Sony (*Figura 2.5c*), un cane robot in grado di riconoscere i comandi dati dal padrone, di vedere e di muoversi in modo autonomo. L'obiettivo per il futuro è far sì che il settore si affermi e raggiunga una rapida espansione, contribuendo all'attuazione della profezia enunciata da Bill Gates, secondo la quale i robot saranno la prossima tecnologia ad invadere le case di tutti, come è avvenuto con i computer e i telefoni cellulari [3].

Anche in AIRLab [4], al Politecnico di Milano, nel corso degli anni sono stati realizzati diversi robogame, classificabili come PIRG (Physically Interactive RoboGame) [5], in cui i partecipanti sono agenti autonomi, di cui almeno uno è un robot e almeno uno è una persona, che interagiscono in un ambiente possibilmente variabile e sconosciuto, seguendo alcune regole, così che i giocatori umani possano divertirsi. Le caratteristiche che li contraddistinguono sono: l'autonomia in ambienti sconosciuti e dinamici, la robustezza fisica, la facilità d'uso, l'affidabilità, la sicurezza e l'elevato livello della qualità dell'interazione. Alcuni dei PIRG creati recentemente in AIRLab sono:

- Jedi Training [6], ispirato ai film Star Wars, in cui un drone volante cerca di colpire l'apprendista Jedi con un raggio laser e quest'ultimo deve difendersi usando la sua spada (*Figura 2.6a*);

- RoboTower [7], ispirato alla tipologia di videogame in cui bisogna difen-

(a) Jedi Training

(b) RoboTower

Figura 2.6: Robogame in AIRLab

dere una torre, nel quale il giocatore deve impedire al robot di distruggere tutte le torri mediante ostacoli e carte con poteri speciali (*Figura 2.6b*);

- Pac-Bot [8], ispirato al famoso videogame Pac-Man, in cui il giocatore può scegliere di rappresentare Pac-Man, raccogliendo i puntini nel labirinto ed evitando i fantasmi nemici, oppure un fantasma, cercando di ostacolare Pac-Man.

Anche da questi pochi esempi risulta evidente come spesso l'ispirazione abbia origine proprio da un videogame esistente, reso concreto e portato nel mondo reale, rispecchiando il percorso evolutivo che ha portato alla nascita dei robogame. Secondo lo stesso principio, abbiamo deciso di prendere spunto da uno dei videogame più noti, Mario Kart Wii [9] (*Figura 2.7*), per realizzare un robogame consistente in una gara tra kart. KartBot, però, va ancora oltre, in quanto richiama anche le tradizionali gare con le macchinine, uno dei giochi più utilizzati nell'infanzia e non solo.

2.2 Le robot racing

La realizzazione concreta di KartBot, in particolare per quanto riguarda lo sviluppo del kart autonomo, ci ha portati a confrontarci con problematiche già affrontate e ampiamente studiate nella realizzazione di veicoli autonomi. La navigazione autonoma costituisce, infatti, un settore che ha già raggiunto risultati concreti e rilevanti anche dal punto di vista applicativo e che riunisce in sé aspetti di diverse discipline, quali la robotica, l'intelligenza artificiale, la

Figura 2.7: Mario Kart Wii

visione artificiale, la meccanica e l'elettronica. Per permettere ad un veicolo di muoversi da solo occorre raccogliere un certo numero di informazioni provenienti da sensori di diverso tipo: da quelli visivi, che servono a riconoscere forme, colori, oggetti e scene, a quelli di prossimità, che servono a rilevare la presenza di ostacoli nelle vicinanze, sia fissi che in movimento. L'unione di tali informazioni è poi utilizzata per prendere decisioni sulla direzione e la velocità a cui il veicolo deve muoversi.

Lo studio dei veicoli autonomi, oltre ad avere scopi commerciali, costituisce un ambito di ricerca in varie università del mondo. In alcuni casi, essa è finalizzata alla partecipazione ad una delle numerose *robot racing* esistenti, competizioni che si svolgono annualmente, spesso a livello internazionale, in cui i veicoli si sfidano a percorrere tracciati di diversa tipologia e difficoltà. Lo scopo principale dell'organizzazione di queste gare è, appunto, quello di incentivare una costante ricerca nel settore, atta ad ottenere tecnologie sempre più avanzate. Le robot racing che si svolgono su pista, in ambienti che presentano una struttura predefinita, sono quelle che, tra tutte, affrontano problematiche più simili a quelle con cui abbiamo dovuto confrontarci nella realizzazione di KartBot, in quanto la navigazione avviene senza la variabilità data da un contesto urbano.

La funzionalità di base da implementare per affrontare una gara su pista è, ovviamente, fare in modo che il veicolo sia in grado di percorrere il tracciato prestabilito senza uscire dai bordi. All'inizio dello studio del settore, questo problema era stato notevolmente semplificato, utilizzando percorsi con pareti, in modo che i veicoli autonomi riuscissero a percepirle, attraverso l'uso di più sensori di prossimità montati lateralmente, e sfruttassero le informazioni

(a) Mappa (b) Tracciato

Figura 2.8: IGVC

sulla loro posizione per muoversi all'interno del percorso. Nella realizzazione di KartBot, adottare tale tecnica ci è sembrato troppo banale e vincolante. Il nostro scopo è, infatti, quello di creare un gioco che sia il più possibile vicino alla realtà e altamente innovativo, affidando la navigazione al sistema di visione. Inoltre, non utilizzare le pareti rende l'ambiente molto più flessibile e con un ingombro più limitato, soprattutto nell'eventualità di modifiche future che permettano di miniaturizzarlo o, volendo, anche di renderlo un puzzle ricomponibile a piacimento, il tutto nell'ottica di una possibile diffusione nelle case delle persone. Per KartBot abbiamo deciso, quindi, di far sì che il kart fosse in grado di percorrere il tracciato utilizzando esclusivamente la differenza di colore. Oggi esistono principalmente tre robot racing che si basano su idee molto simili alla nostra, che si svolgono, però, all'aperto, presentando pertanto notevoli problemi dovuti alla continua variazione di luminosità, a cui i sistemi di visione sono molto sensibili.

In ordine temporale, la prima istituita è stata l'Intelligent Ground Vehicle Competition (IGVC) [10], che si tiene all'Oakland University in Michigan dal 1993 (*Figura 2.8*). La competizione si svolge su un percorso erboso delimitato da linee bianche dipinte sull'erba, al cui interno sono disposti alcuni ostacoli, e il termine del tracciato viene assegnato mediante le sue coordinate GPS.

Per poter affrontare la gara, Scorpion [11] (*Figura 2.9*), il veicolo realizzato dalla California State University di Northridge, vincitore della sfida di navigazione autonoma del 2013, è stato equipaggiato con una videocamera, un sistema SPAN (Synchronized Position Attitude Navigation), reso maggiormente preciso da un DGPS, una bussola, contenente giroscopi ed accelerometri, e degli LRF (Laser Range Finder), sia orizzontali sia verticali, che possiedono

Figura 2.9: Scorpion

un raggio di rilevamento di 8 m. Il percorso da seguire è individuato tramite riconoscimento del colore bianco sull'immagine, mentre gli ostacoli vengono individuati attraverso gli LRF, che effettuano una scansione continua in senso orario. Nel caso in cui la videocamera sia accecata dalla luce del sole, la velocità viene diminuita fino a quando le condizioni ritornano ottimali. Il sistema di localizzazione, altamente preciso, è utilizzato principalmente per la parte della gara in cui il veicolo deve percorrere il tracciato toccando alcuni punti di via assegnati mediante le loro coordinate GPS. Le informazioni provenienti dal sistema di visione e dai sensori vengono infine integrate ed utilizzate per la pianificazione del percorso e la navigazione all'interno della pista. In base ad esse, il veicolo individua il movimento successivo in modo da direzionarsi verso il punto medio dello spazio libero riconosciuto, che sarà ristretto nel caso della presenza di ostacoli. Il costo complessivo del veicolo e delle apparecchiature montate è nettamente superiore a 12000 $, il sistema SPAN e il DGPS sono frutto di una donazione.

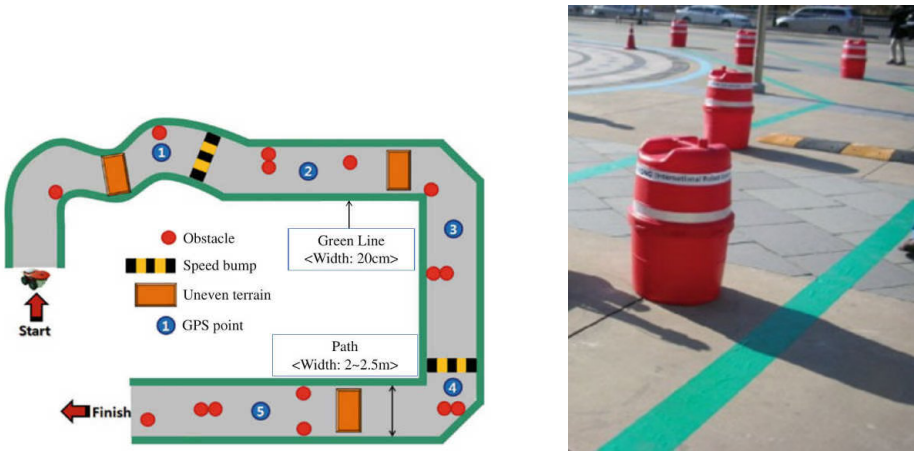
La seconda competizione in questa categoria, l'International Autonomous Robot Racing Challenge (IARRC) [12] (*Figura 2.10*), si è tenuta per la prima volta nel 2005 all'University of Waterloo in Ontario. La gara si svolge su un percorso delimitato da coni di colore arancio e si articola in due aspetti, per quanto riguarda la navigazione: nella Drag Race i concorrenti si sfidano a coppie a percorrere un breve tratto di rettilineo il più velocemente possibile, nella Circuit Race, invece, i concorrenti devono intraprendere un circuito più complesso, rispettando anche segnali di stop e semafori.

Il veicolo realizzato dall'University of Waterloo Autonomous Racing Team, Megalodon [13] (*Figura 2.11*), vincitore della Circuit Race del 2010, sfrutta, per

Figura 2.10: IARRC

la percezione dell'ambiente circostante, due videocamere e un sensore LRF, con un raggio di 30 m e buone prestazioni anche all'aperto in condizioni di elevata luminosità. In particolare, il sistema di visione è utilizzato per estrapolare dalle immagini i coni che costituiscono i bordi della pista, il segnale di stop e il semaforo, mediante l'estrazione in successione di colore, dimensione e forma. Il sensore LRF è usato per la percezione degli ostacoli e, insieme alle videocamere, per creare un sistema ibrido in grado di fornire una maggiore accuratezza nel rilevamento del limite del tracciato. L'elaborazione è effettuata da un computer Samsung SENS Q45, con un processore dual core a 2,1 GHz. Il calcolo della posizione è effettuato mediante un encoder a rotazione montato sulle ruote, un sistema GPS, che può anche non essere utilizzabile nel caso in cui la competizione si tenga all'interno, e un sistema di navigazione inerziale. I veicoli avversari sono individuati quando nelle immagini e nella scansione con l'LRF viene rilevato un oggetto che non è un cono e il sorpasso viene effettuato unicamente in condizioni di estrema sicurezza e solo in alcune zone particolari della pista, in quanto non è prevedibile il comportamento dell'altro concorrente. Per scegliere il movimento successivo da effettuare, il veicolo utilizza un sistema di pianificazione basato sulle mappe di Voronoi create a partire dai coni rilevati, individuando il percorso sicuro meno costoso. La particolarità di Megalodon è che, mentre effettua il primo giro della pista, costruisce una mappa a griglia, marcando la tipologia del percorso incontrata, in modo da poter effettuare i giri successivi più velocemente. In questa mappa vengono anche segnati i punti in cui sono presenti cartelli di stop e semafori, in modo da limitarne poi la ricerca nell'immagine unicamente quando è necessario. Il costo complessivo

Figura 2.11: Megalodon



(a) Mappa

(b) Tracciato

Figura 2.12: IRONC

del veicolo e delle apparecchiature montate è di 24965,00 CAD.

L'ultima competizione indetta è stata l'International Robot Outdoor Navigation Contest (IRONC) [14], che si tiene al Gwangju Technopark in Corea dal 2009 (*Figura 2.12*). La gara si svolge su un percorso delimitato da linee di colore verde acqua della larghezza di 20 cm, all'interno del quale sono presenti ostacoli e zone di terreno sconosciuto. Lungo il percorso vi sono anche dei punti identificati tramite le coordinate GPS, che il veicolo dovrà riconoscere, emettendo un suono.

Il veicolo vincitore della seconda edizione della competizione è stato prodotto dalla Hanyang University della Corea del Sud (*Figura 2.13*) ed è equipaggiato con una videocamera, utilizzata per il riconoscimento delle linee, un

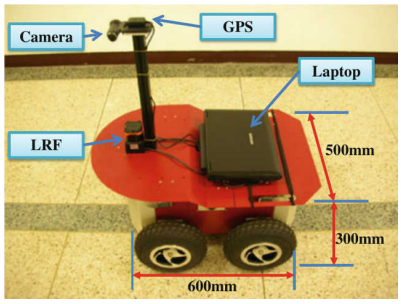


Figura 2.13: Veicolo della Hanyang University

LRF, per la rilevazione degli ostacoli, e un sistema GPS, per la localizzazione dei punti richiesti, che risulta sufficiente in quanto non è richiesta una precisione elevata. L'elaborazione è effettuata da un computer Samsung SENS Q70, con un processore a 2,1 GHz. Il riconoscimento delle linee viene fatto mediante un algoritmo piuttosto robusto di estrazione del colore, in grado di operare anche in condizioni di luminosità molto differenti. Anche in questo caso, le informazioni provenienti dal sistema di visione e dal sensore di prossimità vengono fuse per permettere al veicolo di navigare all'interno della pista in zone sicure, anche in presenza di parti di tracciato non visibili a causa della forte intensità della luce solare. La direzione in cui il veicolo si muove è scelta in base all'angolo esistente tra la posizione attuale e il punto medio del passaggio sicuro rilevato, considerandone il punto in cui esso è più stretto.

In tutte e tre le gare descritte, i vincoli di velocità richiesti ai veicoli partecipanti sono decisamente ridotti, il che permette di analizzare la situazione con molto anticipo e di prendere decisioni con un elevato livello di sicurezza, grazie anche all'utilizzo di telecamere di buona qualità e sensori molto precisi. L'esecuzione in tempo reale, quindi, non costituisce un problema, considerando inoltre che i computer montati a bordo possiedono una capacità computazionale rilevante. Il principale aspetto negativo dell'utilizzo di una strumentazione di livello alto è, però, rappresentato dai costi eccessivamente elevati necessari per la realizzazione complessiva del veicolo. Altro aspetto da considerare è che, ad eccezione della IARRC, in cui comunque la dimensione della pista è molto ampia, nelle altre due competizioni non è prevista la presenza in pista di più di un concorrente in contemporanea all'interno del tracciato, quindi non è richiesta la gestione di ostacoli di tipo dinamico.

(a) Gioco in funzione

(b) Elementi

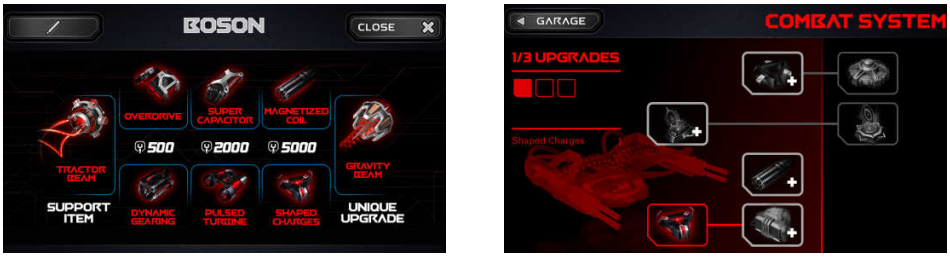
Figura 2.14: Anki Drive

2.3 Anki Drive: racing game per dispositivi Apple

Nel mese di giugno 2013, poco dopo l'inizio del lavoro per la creazione di KartBot, Anki, una startup nata dall'idea di Boris Sofman, Mark Palatucci e Hanns Tappeiner, tre ricercatori nel settore della robotica diplomati al Carnegie Mellon Robotics Institute, ha presentato Anki Drive [15] (*Figura 2.14*), introdotto successivamente sul mercato americano il 23 di ottobre. Anki Drive, definito dai suoi creatori "un videogame nel mondo reale", è un gioco del tutto simile a KartBot e, come questo, costituisce un'esperienza ibrida sospesa tra videogame, giocattolo e robotica. Il gioco ha immediatamente attratto l'interesse di Apple, così come quello dei numerosi investitori, che hanno apportato nella compagnia finanziamenti pari a 50 milioni di dollari.

Anki Drive consiste in una gara tra macchinine dotate di intelligenza artificiale, che le rende in grado di guidarsi da sole, fino a una velocità massima di 1,5 m/s. In pista possono esserci fino a quattro veicoli e i giocatori possono prendere il controllo di uno o più di essi, mediante l'utilizzo di un iPhone, un iPod touch o un iPad. In questo modo è possibile sfidare gli amici o l'intelligenza artificiale stessa, prestando attenzione perché essa è programmata per vincere e cercherà di eliminare gli avversari con ogni mezzo. Per poter cominciare a giocare è sufficiente srotolare la pista, di dimensione 1,07 × 2,6 m, realizzata in vinile nero e lucido, su una superficie piana, in un ambiente chiuso. Essa è venduta insieme a due macchinine che misurano 8,26 × 4,45 cm, fornite di caricabatterie, in un kit iniziale che ha il prezzo di 199,95 $, mentre ulteriori macchinine aggiuntive hanno un costo di 69,95 $ l'una.

Il gioco è reso possibile da un'apposita applicazione per iOS (*Figura 2.15*),



(a) Esempio di schermata (b) Esempio di schermata

Figura 2.15: Applicazione

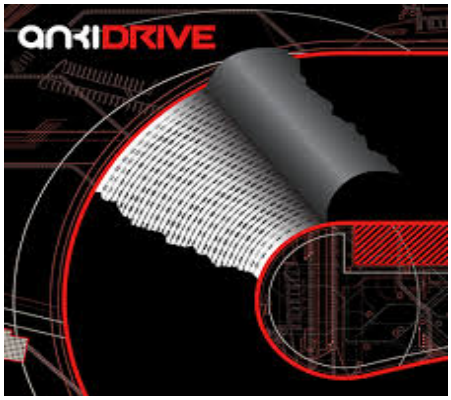
che scambia con le macchinine una grande quantità di informazioni, tramite bluetooth a bassa energia. È proprio questa, infatti, a contenere realmente l'intelligenza artificiale, in grado di controllare ogni aspetto necessario al funzionamento di Anki Drive. L'idea è quella di avere un software che funge da cervello di un sistema complesso, costituito da robot fisici con conoscenze e capacità limitate, rendendoli in grado di costituire, presi nel loro insieme, qualcosa di altamente intelligente.

Nel gioco esistono quattro tipologie di macchinine, ognuna con abilità proprie, caratteristiche diverse di accelerazione, velocità ed energia e con specifici equipaggiamenti virtuali (*Figura 2.16*). Il loro utilizzo permette di guadagnare punti, da poter riscattare e sfruttare per farle evolvere, migliorandone gli aspetti relativi a motore, telaio, armi e difese per il combattimento ed energia. Tutto ciò è permesso dall'applicazione, che mantiene all'interno del telefono uno stato virtuale, con la "personalità" delle auto: le modifiche non portano cambiamenti fisici, ma semplicemente un diverso comportamento, definito dal software.

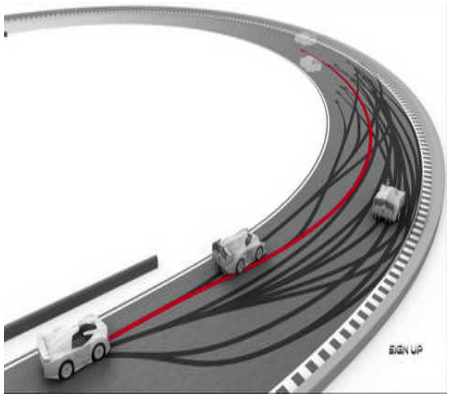
Anki Drive potrebbe all'apparenza sembrare semplicemente una corsa di macchinine, ma in realtà si tratta di una vera e propria battaglia. Non vince chi arriva primo, ma chi raggiunge per primo il numero di punti prestabilito, colpendo gli avversari con le armi a disposizione. Esse sono delle tipologie più disparate: da quella che fa girare la macchinina e la fa guidare contro il flusso del traffico, alla frenata brusca che fa volare l'avversario dietro di essa, all'impulso elettromagnetico che spara un raggio esplosivo, al semplice clacson che spinge le altre macchinine fuori dal propria traiettoria. Il feedback dato dall'utilizzo delle armi sull'avversario è costituito da luci e flash sui veicoli e da effetti sonori generati dal dispositivo che esegue l'applicazione.

Figura 2.16: Abilità delle macchinine

L'hardware delle macchinine è costituito da componenti poco costosi, che comprendono due motori, una batteria, un microcontrollore e un sensore di tracciamento, ossia un lettore a infrarossi, posto tra le ruote davanti, rivolto verso il basso. Esso è in grado di percepire le informazioni integrate nella pista, coperta da uno speciale pattern, simile alla traccia di un disco in vinile (*Figura 2.17a*). Utilizzando un processore da 50 MHz, le macchinine leggono tale codice, che fornisce loro la conoscenza della rispettiva posizione sul tracciato e trasmettono all'applicazione l'informazione ottenuta. Essa, sfruttando la potenza di calcolo del dispositivo mobile, utilizzerà tali informazioni per calcolare, con algoritmi molto avanzati, la traiettoria da seguire e i sorpassi e le frenate da effettuare, in base ad una ricerca approfondita di quello che la macchinina dovrà fare e di quello che faranno gli avversari. L'intero rilevamento delle collisioni è, infatti, gestito dall'applicazione, in quanto le auto non sono in grado di vedersi tra loro. L'applicazione analizza, con una frequenza pari a 500 volte al secondo, migliaia di potenziali azioni, ottenendo una pianificazione migliore di una semplice reazione istantanea. Il principio su cui il suo funzionamento si basa è ispirato a quello dei robot inseguitori di linee. In questo caso, però, la linea non esiste davvero: il sofisticato software crea qualunque manovra si desideri e la converte in una linea virtuale che la macchinina segue come se si trattasse di una linea fisica (*Figura 2.17b*). È all'interno del dispositivo mobile che ha luogo tutta la computazione e vengono prese le decisioni: alle auto spetta solo il compito di eseguire i comandi ricevuti. L'applicazione, quindi, non rappresenta un semplice controllo remoto, ma ciò che conduce l'esperienza,



(a) Struttura del tracciato



(b) Calcolo della traiettoria

Figura 2.17: Pista

attraverso un utilizzo dei dispositivi diverso da quello finora sperimentato.

Ciò che, però, ha lasciato perplessi i fortunati che hanno già avuto la possibilità di provare Anki Drive, è proprio il fatto che anche le automobiline sotto il controllo dei giocatori si guidano da sole. I giocatori assumono semplicemente un ruolo di assistenti alla guida, in grado di effettuare solo alcune intromissioni durante il gioco: decidere quanto veloce la macchinina deve andare, farla spostare a destra e a sinistra per costringerla a fare una curva più larga o più stretta e utilizzare le armi a disposizione. In pratica, anche le macchinine sotto il controllo del giocatore stanno da sole in pista e sono in grado di compensare, nei limiti del possibile, le sue manovre errate. Anche nel caso in cui si verifichino piccoli urti o giri su se stessa, l'auto è in grado di riportarsi in carreggiata e continuare la gara. Nel caso di collisioni maggiori o che lascino la macchinina sottosopra, invece, il giocatore viene avvisato di ripristinare la situazione corretta, riposizionandola sul tracciato. Inoltre, il gioco risulta semplice da giocare, ma più complesso da padroneggiare, a causa dell'elevata velocità delle auto. Secondo alcune statistiche sui primi mesi di utilizzo, infatti, nel caso in cui sia stata selezionata la difficoltà più avanzata, l'intelligenza artificiale ha vinto in media in 9 casi su 10. Altre critiche avanzate sono state relative alla breve durata della carica della batteria, di soli 20 minuti, che richiede poi un tempo di caricamento tra gli 8 e i 10 minuti, e all'impossibilità di riconoscere ostacoli che non siano macchinine sul tracciato. La limitazione principale resta comunque la totale mancanza di flessibilità, dovuta in primo luogo all'obbliga-

torietà di possedere un dispositivo Apple per giocare. Inoltre, la disponibilità di un'unica configurazione della pista, può comportare una rapida perdita di interesse dei giocatori.

Capitolo 3

Descrizione del gioco

3.1 Descrizione del gioco e regolamento

KartBot è un robogame ispirato al famoso videogame Mario Kart Wii. Il gioco consiste in una gara tra due kart, caratterizzata dalla possibilità di ottenere ed utilizzare armi bizzarre per ostacolare l'avversario e ottenere così la vittoria. In KartBot i kart sono in realtà robot su ruote, uno dei quali è controllato dal giocatore mediante un Wii Remote [16] e l'altro dall'intelligenza artificiale oppure, se lo si desidera, da un'altra persona.

Abbiamo scelto di utilizzare il controller Wii Remote proprio al fine di mantenere l'analogia con Mario Kart Wii. I tasti da utilizzare per il gioco sono soltanto tre: il tasto "1", che se premuto permette di accelerare e se rilasciato di decelerare; il tasto "2", che ferma istantaneamente il kart; il tasto "B", che permette di utilizzare l'arma, se presente. Per sterzare è sufficiente inclinare il controller.

Per il funzionamento di KartBot è necessario disporre di un computer sul quale eseguire l'applicazione del gioco, posizionato in modo che sia ben visibile durante tutta la gara. Prima della partenza i kart devono essere collocati nell'apposito spazio contrassegnato sulla pista e dall'applicazione va indicata la loro posizione di partenza. Allo scattare del semaforo la gara ha inizio. A questo punto, per il kart controllato, sullo schermo vengono visualizzati lo streaming di quello che vede durante la gara e le informazioni importanti per essa: la sua posizione attuale in classifica, il giro corrente e l'arma di cui dispone in quel momento.

Le armi appartengono a cinque diverse tipologie e sono classificabili in

vantaggiose e svantaggiose. Le armi svantaggiose sono quelle che colpiscono il kart nemico, al fine di procurargli una penalizzazione. Vi sono tre tipi di armi svantaggiose:

- *Fulmine*: ferma il kart davanti per 3 secondi e quando riparte va alla velocità minima per altri 3 secondi. Inoltre fa perdere l'eventuale arma in possesso (*Figura 3.1a*);

- *Stop*: ferma il kart davanti per 3 secondi e gli fa perdere l'eventuale arma in possesso (*Figura 3.1b*);

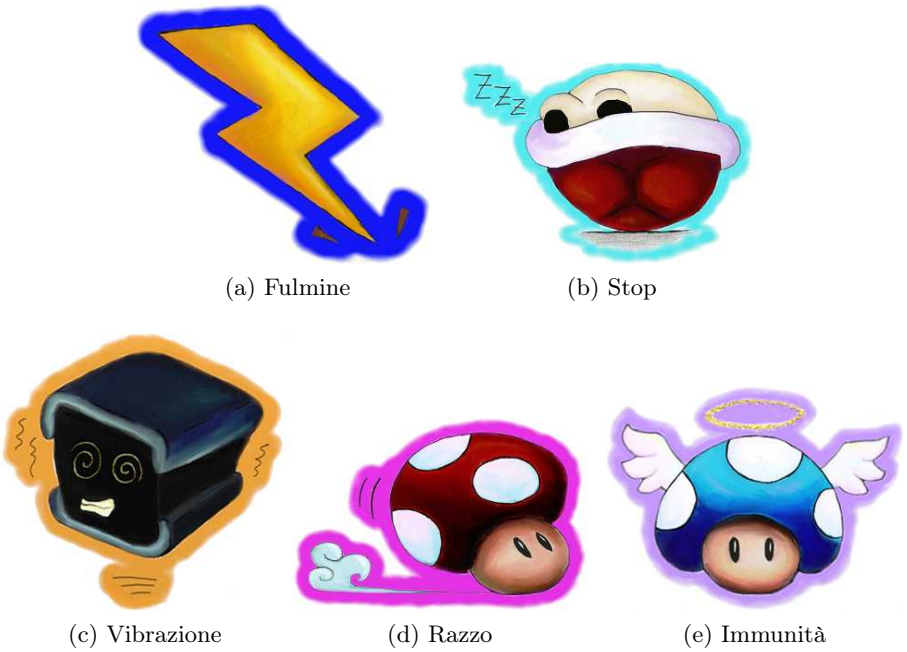
- *Vibrazione*: fa procedere a scatti il kart davanti per 7 secondi (*Figura 3.1c*).

Le armi vantaggiose, invece, sono quelle che creano una situazione favorevole al kart che le utilizza. Ve ne sono di due tipi:

- *Razzo*: fa andare più veloce il kart che la utilizza per 3 secondi (*Figura 3.1d*);

- *Immunità*: rende il kart che la utilizza immune al primo utilizzo di un'arma di tipo stop o vibrazione. Nel caso di utilizzo di un'arma fulmine, invece, l'immunità è inefficace (*Figura 3.1e*).

Le armi sono ottenute passando con il kart sopra uno degli indicatori a forma di stella presenti sulla pista (*Figura 3.2*), i quali, non essendo disattivati da tale evento, permettono, eventualmente, all'avversario di usufruirne immediatamente. La tipologia viene assegnata in base alla posizione occupata in quell'istante: le armi di tipo fulmine, stop, vibrazione e razzo sono disponibili per il giocatore in posizione arretrata e hanno la stessa probabilità di venire assegnate, mentre per il giocatore in prima posizione è disponibile unicamente l'immunità, per permettergli di difendersi. Nel caso di più armi agenti in contemporanea su di un kart, l'effetto totale è dato dalla somma degli effetti delle armi agenti. Riutilizzare, invece, la stessa arma, nel caso essa stia già agendo, non provoca modifiche nella durata, quindi viene sprecata. Il momento in cui l'arma acquisita viene utilizzata è scelto dall'utente tramite Wii Remote nel kart controllato e dall'intelligenza artificiale nel kart autonomo.



(a) Fulmine (b) Stop

(c) Vibrazione (d) Razzo (e) Immunità

Figura 3.1: Armi

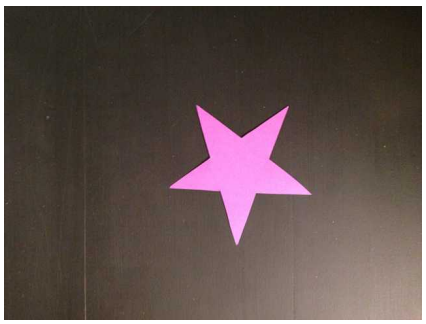


Figura 3.2: Indicatore arma

La durata della gara è pari a 4 giri. Il giocatore che per primo taglia il traguardo vince e la classifica finale viene visualizzata sullo schermo.

Il regolamento del gioco è semplice e anche piuttosto scontato, in quanto si basa unicamente su criteri di correttezza e di onestà:

1. Non è possibile urtare l'avversario al fine di danneggiarlo. Qualsiasi urto, anche involontario, causerà l'immediata interruzione manuale della gara;

2. Non è possibile uscire dalla pista al fine di avvantaggiarsi. Se fuoriesce dal bordo, il kart rallenta alla velocità minima e il Wii Remote restituisce

al giocatore un'esperienza di gioco, mediante la vibrazione;

3. Il mancato passaggio dalla linea del traguardo rende il giro non valido.

KartBot è pensato in modo da risultare semplice e giocabile da chiunque, essendo intuitivo e basato su regole che sono ovvie ed intrinseche in qualunque gara con le macchinine. L'utilizzo del controller Wii Remote è finalizzato proprio a questo scopo, in quanto i movimenti da effettuare sono molto simili a quelli della guida di un'automobile. Il tutto è notevolmente migliorato, inoltre, dall'aggiunta di meccanismi di feedback per l'utente, come l'accensione di alcuni led per indicare il possesso di un'arma e l'utilizzo di suoni durante il gioco: il conto alla rovescia prima dell'inizio della gara, le espressioni di soddisfazione o disappunto durante i sorpassi e l'utilizzo delle armi e la musica finale per celebrare la vittoria. KartBot è, in sintesi, ideato con l'obiettivo di fornire la massima esperienza di gioco possibile. In *Appendice D* abbiamo riportato il manuale utente del gioco.

3.2 Funzionalità del kart

In KartBot entrambi i kart sono costituiti da robot su ruote. Dei due, quello controllato dall'intelligenza artificiale è sicuramente il più interessante, in quanto esso, essendo totalmente autonomo, necessita di prendere tutte le decisioni basandosi esclusivamente sulle informazioni ricevute da ciò che lo circonda. In presenza di un ambiente dinamico, come quello del gioco, la quantità di percezioni raccolte e da analizzare è elevata e va gestita velocemente. Per implementarlo, abbiamo scelto di utilizzare un'architettura basata su un approccio di tipo reattivo, in cui non è presente un modello del mondo, ma solo di alcuni aspetti di esso. I dati sensoriali raccolti durante il movimento influenzano i diversi moduli dell'architettura, corrispondenti alle funzionalità implementate, al fine di generare mediante essi le azioni necessarie. Le funzionalità di cui i kart necessitano per lo svolgimento del gioco, come si può vedere in *Figura 3.3*, sono:

- *gestione della navigazione:* è responsabile di tutti gli aspetti relativi al movimento del kart all'interno della pista. Si occupa di riconoscere la forma del tracciato, di evitare il kart avversario e di localizzare e provare a raggiungere le armi. In base alle informazioni acquisite dall'analisi

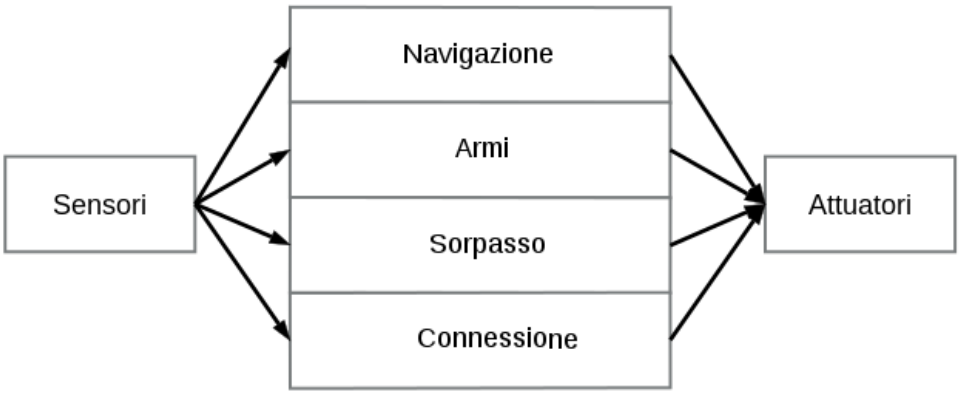


Figura 3.3: Approccio reattivo

delle immagini, essa calcola la traiettoria ed esegue l'azione successiva, in termini di angolazione dello sterzo e velocità dello spostamento;

- *gestione del sorpasso:* si occupa dell'identificazione del sorpasso da parte di un altro kart e, nel caso esso sia avvenuto, dell'invio della notifica all'applicazione per l'aggiornamento della classifica attuale. Il sorpasso è di tipo passivo, cioè è il kart sorpassato che si occupa di comunicare il verificarsi dell'evento. La scelta tra rilevazione del sorpasso di tipo attivo e di tipo passivo, apparentemente arbitraria, è stata pensata in previsione di un'eventuale espansione futura, che comporti la presenza di più kart in pista. In tal caso, infatti, se il sorpasso è multiplo, per il kart sorpassato è possibile comunicare la presenza di un altro concorrente sul lato opposto, in modo da aggiornare correttamente le posizioni correnti, cosa non possibile se la rilevazione del sorpasso avvenisse in modo attivo;
- *gestione delle armi:* è responsabile della rilevazione dell'acquisizione di una nuova arma, della selezione della tipologia di essa, in base alla posizione occupata, e della notifica di ciò all'applicazione. Inoltre, si occupa anche di notificare l'utilizzo dell'arma in possesso e di realizzare gli effetti delle armi agenti in quel momento sul kart;
- *gestione della connessione:* ha lo scopo di comunicare con l'applicazione al fine di implementare le dinamiche di gioco.

Di queste, la gestione del sorpasso, delle armi e della connessione sono utilizzate da entrambi i kart, quella della navigazione, invece, è caratteristica del kart autonomo. Essa costituisce di sicuro la componente di maggiore rilevanza dal punto di vista ingegneristico. L'aspetto innovativo è rappresentato dal fatto che il kart si muove in un ambiente dinamico ad elevata velocità, senza riferimenti fissi o utilizzo di modelli, ma basandosi unicamente sulla visione. Inoltre, un altro aspetto importante da considerare, è che tutta l'elaborazione necessaria ad essa avviene totalmente a bordo del kart, in tempo reale.

3.3 Caratteristiche dell'ambiente di gioco

L'ambiente di gioco è rappresentato da una pista della dimensione di 5,52 × 3,68 m, una grandezza adatta ad una gara avvincente. Il tracciato ha una larghezza di 72 cm e il suo interno è di colore nero, mentre l'esterno è verde. La dimensione della larghezza è frutto di un compromesso tra la necessità di visualizzarne con la videocamera una porzione sufficiente per l'analisi e quella di permettere alle macchinine di allinearsi in tutta sicurezza durante un sorpasso. La forma da noi realizzata è mostrata in *Figura 3.4*, ma essa non rappresenta un vincolo, in quanto non è in alcun modo predefinita nel codice, permettendo quindi eventuali modifiche future. Anche i colori stessi possono essere facilmente sostituiti, a patto di modificare i valori delle soglie utilizzate per il loro riconoscimento nell'immagine, la cui ricerca può essere effettuata mediante il tool di taratura che abbiamo appositamente creato.

La scelta del materiale con cui realizzare la pista è stata soggetta a diverse prove e ad un'accurata valutazione dei pro e dei contro. La prima prova è stata fatta utilizzando pezzi di cartone, che sono apparsi subito troppo facilmente deteriorabili. La prova successiva è stata effettuata sfruttando un telo di materiale plastico, che presentava però un problema importante: la necessità di piegare la pista per trasportarla lasciava su di esso delle curvature non più eliminabili, che impedivano il corretto riconoscimento della forma del tracciato. Per la realizzazione abbiamo scelto, quindi, di utilizzare un pavimento laminato in legno, in modo che risultasse di aspetto gradevole, perfettamente liscio e senza discontinuità. Anche questa soluzione non è comunque ottimale, a causa del tempo e dell'attenzione richiesti per il montaggio, ma i vantaggi sono nettamente predominanti. Il pavimento in laminato è costituito da listoni dotati di appositi incastri per l'assemblaggio, in modo da poterli smontare in caso

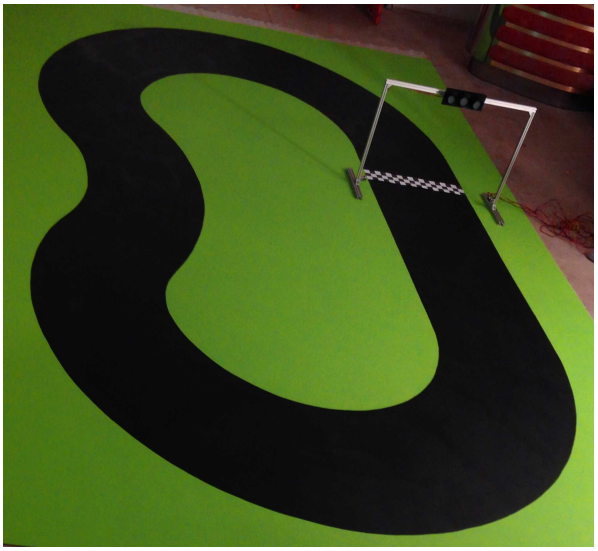


Figura 3.4: Pista

di trasporto. Ciò presenta anche il vantaggio di poter sostituire con facilità eventuali pezzi danneggiati. Per la realizzazione della pista abbiamo innanzitutto carteggiato i pezzi per aumentare l'aderenza della vernice e, dopo aver realizzato il disegno, abbiamo applicato uno smalto molto opaco, in modo da limitare i problemi dati da intensi riflessi di luce. Una corretta illuminazione rappresenta, infatti, un elemento importante, a causa dell'elevata sensibilità dei sistemi di visione a variazioni delle condizioni ambientali. In particolare, è necessaria un'illuminazione diffusa, senza luci particolarmente forti che riflettano in modo diretto sulla pista.

Anche la realizzazione grafica del bordo ha subito un'evoluzione. Nelle prime prove la pista era disegnata su un fondo bianco e ciascuno dei due bordi era costituito da due linee affiancate spesse circa 2,5 cm ciascuna, quella dal lato del tracciato nera e quella verso l'esterno arancione, allo scopo di discriminare nell'immagine di quale dei due bordi si trattasse, basandosi sulla posizione dei colori. Tale soluzione è stata poi migliorata esteticamente, in modo da giungere all'aspetto attuale, decisamente più realistico.

La postazione di partenza dei kart è evidenziata da apposite linee, disegnate sulla pista in posizione arretrata rispetto alla struttura metallica ad arco, realizzata sia per l'alloggiamento del semaforo sia per la rilevazione del passaggio dei kart. Tale struttura, visibile in *Figura 3.5,* ha una dimensione sufficientemente grande da non costituire un ingombro per il transito di questi e al di

Figura 3.5: Struttura del traguardo

sotto di essa è disegnata la scacchiera, che rappresenta la linea del traguardo.

Per la rappresentazione delle armi abbiamo scelto di utilizzare delle sagome a forma di stella di colore viola e, in base alla dimensione della pista, ne abbiamo posizionate quattro, suddividendole in due diverse zone del tracciato.

Tutte le scelte fatte per l'ambientazione sono pensate per garantire al giocatore la migliore esperienza di gioco possibile.

Capitolo 4

Architettura hardware

L'architettura hardware del kart è stata realizzata seguendo un approccio modulare e usando componenti a basso costo, facilmente reperibili sul mercato. Essa è pensata in modo da permettere che tutta l'elaborazione avvenga a bordo, senza l'utilizzo di un computer remoto, necessario solo per implementare le funzionalità del gioco. Le interconnessioni logiche ed i componenti utilizzati sono mostrati in *Figura 4.1*.

4.1 Funzionalità

Di seguito abbiamo elencato le funzionalità del kart, per ognuna della quali abbiamo descritto i componenti necessari per la realizzazione.

4.1.1 Navigazione

4.1.1.1 Unità di elaborazione

L'unità di elaborazione è il componente che implementa l'intelligenza artificiale necessaria al corretto funzionamento del kart. Considerando le dimensioni di questo, abbiamo avuto la necessità di scegliere un computer di grandezza ridotta e dal consumo energetico minimale, ma allo stesso tempo sufficientemente performante per i nostri scopi. Attualmente la piattaforma più richiesta e diffusa è Raspberry Pi [17] (*Figura 4.2b*), un mini computer prodotto dalla Raspberry Pi Foundation, equipaggiato con un processore ARM1176JZF-S a 700 MHz e 512 MB di RAM. Dopo aver effettuato i primi test, ci siamo resi conto che esso non era adeguato alle nostre necessità: le immagini venivano

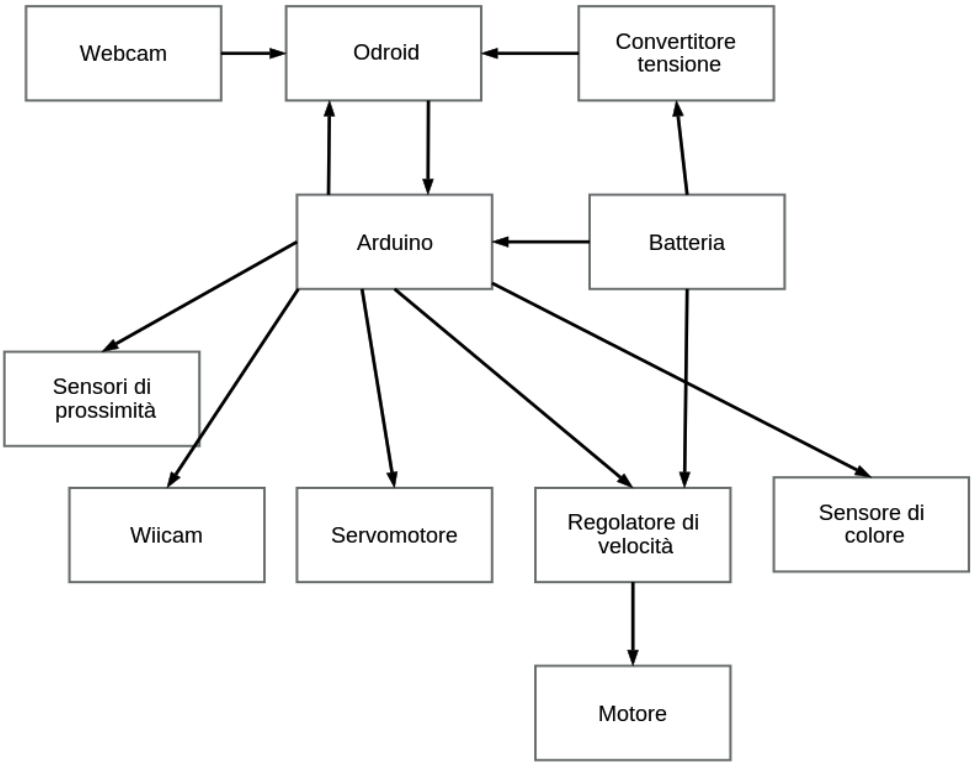
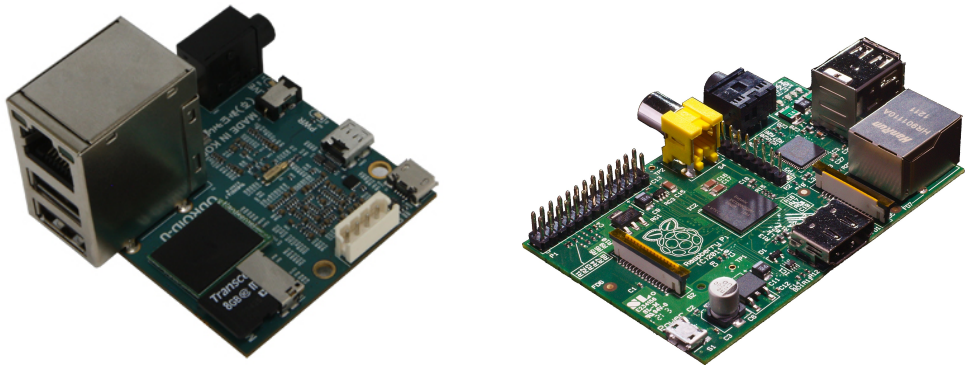


Figura 4.1: Architettura hardware

acquisite ad una frequenza di circa 5 per secondo e, inoltre, il loro tempo di elaborazione non era sufficientemente basso.

Considerando che il kart deve presentare un comportamento altamente reattivo, abbiamo ricercato una piattaforma più performante rispetto a Raspberry Pi. La nostra scelta è ricaduta su Odroid U2 [18] (*Figura 4.2a*), un mini computer prodotto dall'azienda sud coreana Hardkernel, equipaggiato con un processore Quad core ARM Cortex-A9 MPCore a 1,7 GHz e 2 GB di RAM. Con esso abbiamo rilevato come la frequenza con cui venivano acquisite le immagini, a parità di webcam, fosse di circa 25 per secondo ed il tempo di elaborazione di una di queste si fosse ridotto a circa un quarto. Tuttavia, dati i costi e i tempi di spedizione più contenuti di Raspberry Pi rispetto ad Odroid, abbiamo deciso di utilizzarlo come unità di elaborazione per il kart controllato tramite Wii Remote, in quanto in esso non è necessaria alcuna analisi sulle immagini. Su entrambe le piattaforme abbiamo installato ArchLinux [19] ARM, data la sua semplicità di utilizzo, il ridotto consumo di risorse e il suo costante



(a) Odroid (b) Raspberry

Figura 4.2: Unità di elaborazione

aggiornamento. In *Appendice A* abbiamo riportato le istruzioni per la loro configurazione.

Entrambe le unità di elaborazione adottate sono dotate di due porte USB, ma noi necessitavamo di quattro di esse per collegare tutti i componenti. A tale scopo, abbiamo equipaggiato entrambi i mini computer con un hub USB a quattro porte, il cui collegamento ha rappresentato un'operazione delicata. Sia Odroid sia Raspberry presentano, infatti, una corrente limitata in uscita su tali porte, che potrebbe risultare insufficiente per alimentare tutte le periferiche collegate, causando un loro funzionamento non corretto. Abbiamo quindi alimentato l'hub USB direttamente dalla batteria di bordo, così che i dispositivi ad esso collegati non assorbano corrente direttamente dall'unità di elaborazione.

4.1.1.2 Sensori

L'acquisizione delle immagini avviene attaverso una webcam Logitech C270 [20] (*Figura 4.3a)*, con una risoluzione HD a 720p e con un angolo di vista di circa 60°, collegata direttamente all'unità di elaborazione tramite USB. Su di essa abbiamo installato una lente che permettesse di aumentare il suo angolo di vista. Al fine di ridurre l'elaborazione sulle immagini e la loro occupazione in memoria centrale, abbiamo impostato che vengano acquisite ad una dimensione di 320 × 240 pixel: questa scelta non causa perdita di informazione, ma offre vantaggi in termini di prestazioni.

Per individuare la presenza di un avversario ai lati del kart abbiamo deci-

so di installare dei sensori di prossimità Sharp 2D120X [21] (*Figura 4.3b),* in grado di rilevare la presenza di oggetti da 4 a 30 cm, con un raggio d'azione di circa 10^{o}. Ognuno di essi è collegato direttamente alla scheda Arduino [22], dalla quale preleva l'alimentazione necessaria al funzionamento, e invia un segnale rappresentante la grandezza misurata su un pin analogico. Non si tratta, comunque, di sensori di precisione ed essi non presentano una caratteristica lineare, ma il costo contenuto e le nostre esigenze non particolarmente elevate, li rendono adatti alla situazione. Per far sì che la misura effettuata sia più stabile e qualitativamente migliore, abbiamo collegato dei condensatori in parallelo ai loro pin di alimentazione. Ogni kart è equipaggiato con quattro di questi sensori, due per lato, uno posto anteriormente ed uno posteriormente, in corrispondenza delle quattro ruote. Sul kart autonomo abbiamo avuto la necessità di aggiungerne altri due in posizione frontale, in modo che il loro raggio d'azione ricada sul punto cieco creatosi tra la fine dell'immagine vista dalla webcam e il sensore di prossimità laterale anteriore.

Come descritto nelle regole, quando un kart esce dal tracciato, deve rallentare per non essere avvantaggiato rispetto agli altri. Per stabilire se esso sia attualmente in pista, abbiamo scelto di equipaggiarlo con il sensore di colore ADJD-S311 [23] (*Figura 4.3c)*, montato sotto la scocca in posizione centrale. Quest'ultimo comunica con Arduino, sfruttando la comunicazione I2C, e da esso preleva l'alimentazione necessaria al suo funzionamento. La scheda di Arduino prevede due pin speciali per tali comunicazioni: SDA e SCL, i quali variano a seconda della versione utilizzata (nel caso di Arduino micro sono i pin digitali 2 e 3). Dal momento che la pista è formata da due colori ben diversi nello spazio RGB, non abbiamo avuto bisogno di identificare esattamente quello del terreno su cui il kart si sta muovendo: la situazione "fuori pista" viene individuata semplicemente quando esso varia in modo significativo. Questo tipo di sensore necessita di essere calibrato ad ogni accensione, tramite il riconoscimento del colore bianco.

Le strategie di gara, usate dal kart autonomo per effettuare un sorpasso, sono subordinate alla corretta identificazione dell'avversario davanti. Inizialmente avevamo previsto di costruire i kart in modo tale che avessero la parte posteriore di forma rettangolare, di colore predefinito, riconosciuto tramite elaborazione sulle immagini. Dopo vari esperimenti, abbiamo notato come questa tecnica non fosse adatta ai nostri scopi, a causa degli eccessivi riflessi, che non permettevano di riconoscere perfettamente il colore, e alle immagini non sempre

(a) Webcam C270

(b) Sensore prossimità

(c) Sensore colore

Figura 4.3: Sensori

nitide, dovute sia al movimento sia alle vibrazioni, le quali non permettevano di riconoscere la forma rettangolare. Come prima alternativa, abbiamo posizionato sulla parte posteriore dei kart dei marker predefiniti, sfruttando le implementazioni contenute nella libreria ARToolkit. Per motivazioni analoghe alle precedenti abbiamo escluso questo metodo.

La scelta definitiva è ricaduta sull'utilizzo del sensore WiiCam [24] (*Figura 4.4)*, una fotocamera IR contenuta nel Nintendo Wii Remote. Nel suo habitat naturale essa riesce ad individuare le sorgenti di infrarosso presenti nella WiiBar per stabilire, tramite semplici calcoli trigonometrici, la posizione del Wii Remote rispetto ad essa. Nel nostro scenario, abbiamo installato una WiiBar nella parte posteriore del kart controllato, che viene individuata dalla WiiCam installata sul kart autonomo. Tramite un apposito circuito elettronico (riportato in *Appendice B*), la WiiCam invia ad Arduino le coordinate di un massimo di quattro sorgenti di infrarosso individuate. Anche in questo caso, la comunicazione con la scheda Arduino avviene tramite bus I2C, sul quale possono essere installati fino a 112 dispositivi, identificati ciascuno da un indirizzo

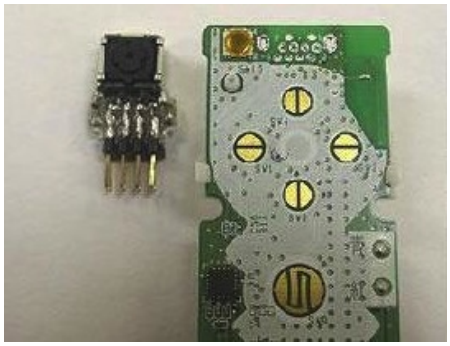


Figura 4.4: WiiCam

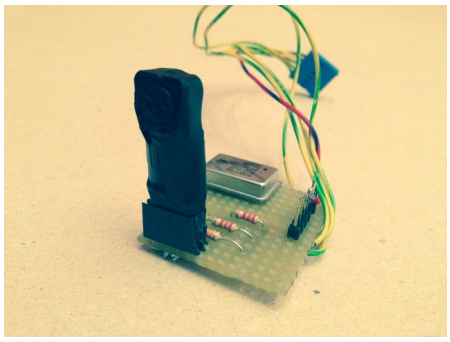


Figura 4.5: Circuito WiiCam

univoco. Di questi uno si elegge come master, qui rappresentato da Arduino, mentre tutti gli altri si identificano come slave, nel nostro caso il sensore di colore e la WiiCam. Da alcune ricerche è emerso che la WiiCam può produrre risultati errati se colpita dalla luce naturale, in quanto in essa vi è la presenza di raggi infrarossi. Tuttavia, essendo il kart posizionato sul pavimento, questa situazione si verifica con scarsa probabilità. Il circuito prodotto è mostrato in *Figura 4.5.*

Il Wii Remote, utilizzato dal giocatore per controllare il suo kart, si collega via bluetooth all'unità di elaborazione presente su di esso, la quale traduce i comandi ricevuti in messaggi da inviare ad Arduino. Tale vincolo ha imposto, quindi, che ogni unità di elaborazione fosse equipaggiata con un adattatore bluetooth USB.

4.1.1.3 Attuatori

Per assicurare il movimento del kart abbiamo utilizzato due tipi di attuatori: un servomotore da 13 g, responsabile dello sterzo delle ruote anteriori, e un motore brushless 2430-4800 kv, comandato da un regolatore di velocità @ 25 A. Entrambi sono collegati direttamente ad un pin digitale di Arduino per ricevere il segnale, mentre il regolatore di velocità preleva l'alimentazione direttamente dalla batteria ed il servomotre dal regolatore di tensione a 5 V. Quest'ultima scelta è data dalla necessità di non sottoporre Arduino ad eccessivo lavoro di conversione di tensione, dato il consumo di circa 200 mAh del servomotore.

4.1.2 Armi

L'aspetto che rende il gioco più avvincente, favorendo la competizione, sono le armi, che vengono acquisite al passaggio del kart sopra una sagoma a forma di stella di colore viola. Inizialmente gli indicatori erano stati realizzati tramite tag RFID, equipaggiando i kart con dei lettori appositi, ma abbiamo notato che, a causa della velocità del movimento, il passaggio sopra di essi non veniva rilevato. Dopo alcuni esperimenti, abbiamo deciso di realizzarli con del cartoncino di colore viola, posizionato sul tracciato dopo uno spessore di plexiglass di 6 mm, come mostrato in *Figura 4.6,* equipaggiando i kart con degli interruttori, posti sotto la scocca nella zona anteriore. Date le dimensioni è stato necessario installare due interruttori per ogni kart, collegati direttamente ad un ingresso digitale di Arduino tramite un circuito interrotto; esso è stato costruito in modo che Arduino riceva su tale pin un livello digitale alto al momento della chiusura del circuito, vale a dire del passaggio del kart sopra il tag, ed un livello digitale basso in ogni altra situazione. Per fare in modo che ciò sia sempre vero e non esistano momenti in cui l'ingresso sia indefinito, abbiamo realizzato una rete di pull down a valle dell'interruttore.

L'acquisizione di un'arma viene segnalata all'utente sia tramite la nostra applicazione sia tramite l'accensione di una serie di led a bordo del kart, in modo da migliorare l'esperienza di gioco. I led sono collegati ad un ingresso digitale di Arduino e vengono accesi immediatamente dopo la pressione dell'interruttore e spenti all'utilizzo dell'arma.

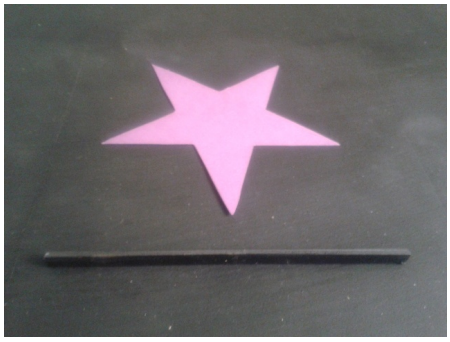


Figura 4.6: Indicatore arma

4.1.3 Sorpasso

Questa funzionalità è responsabile di aggiornare la classifica della gara e, quindi, di stabilire qual è il vincitore. Ogni kart individua l'evento "sorpassato" sfruttando i dati provenienti dai sensori laterali, mediante l'identificazione di due fasi distinte: la prima, in cui il kart riconosce di avere un avversario affiancato, cioè entrambi i sensori di prossimità sullo stesso lato rilevano un ostacolo, e la seconda, in cui il sensore laterale posteriore (dello stesso lato) non lo percepisce più, mentre in quello anteriore è ancora presente. Appena un kart riconosce di essere stato sorpassato invia un messaggio di notifica all'applicazione principale, che aggiorna la classifica e invia in broadcast a tutti i kart la loro nuova posizione aggiornata.

4.1.4 Connessione

Al fine di permettere ai kart di scambiare messaggi con l'applicazione, sia per l'implementazione delle dinamiche di gioco sia per lo streaming video, abbiamo configurato una rete WiFi. A tale scopo, abbiamo utilizzato un access point wireless e equipaggiato tutti i kart con una scheda di rete WiFi connessa al bus USB.

In particolare, lo streaming video, usato per arricchire l'interfaccia di gioco, avviene acquisendo immagini dalla webcam montata a bordo, che prima dell'invio vengono compresse in formato JPEG riducendone la dimensione di circa un terzo, in modo da diminuire l'uso delle risorse di rete, che altrimenti avrebbe potuto causare ritardi nella comunicazione di altre informazioni importanti.

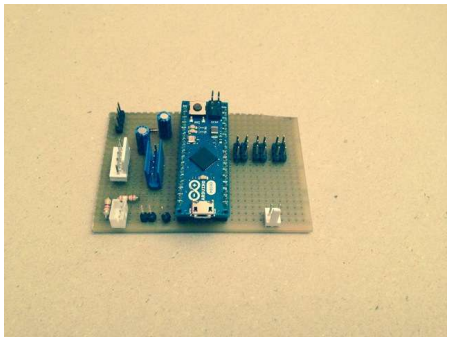


Figura 4.7: Scheda di bordo

4.1.5 Alimentazione

L'energia necessaria a tutti i componenti è fornita da una batteria LiPo 3S 3000 mAh 7,4 V, la quale garantisce un funzionamento continuo di circa un'ora. Questo ci è sembrato un giusto compromesso: una batteria di capacità superiore avrebbe comportato dimensioni fisiche maggiori, non adatte ai kart creati.

La tensione di alimentazione da fornire alle unità di elaborazione, all'hub ed al servomotore è di 5 V, mentre la batteria presenta una differenza di potenziale di 7,4 V. Per trasformarla a 5 V abbiamo inserito un regolatore di tensione detto UBEC a valle della batteria in grado di fornire 5 V stabili con un massimo di 8 A.

In *Appendice B* è riportato lo schema circuitale con tutti i componenti installati, il cui risultato è mostrato in *Figura 4.7.*

4.2 Struttura del kart

La forte ispirazione al gioco Mario Kart Wii ha imposto che il robot avesse l'aspetto di un kart. La costruzione del telaio ci ha posto di fronte a una scelta fondamentale: costruirlo ad-hoc o usare un modellino preesistente. Data la nostra scarsa esperienza nel campo, le possibili difficoltà e il poco tempo a disposizione, abbiamo deciso di acquistarne uno. La prima scelta era ricaduta su Iracer [25], un modello equipaggiato con due motori passo-passo, in grado di implementare la cinematica di un'automobile, e di una scheda di controllo, capace di ricevere comandi tramite bluetooth. Ci siamo resi conto che questo non era adatto ai nostri scopi, in quanto lo sterzo, essendo gestito da un motore

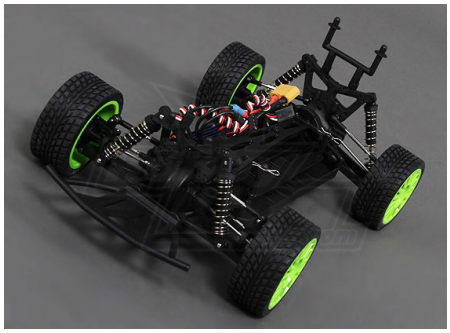


Figura 4.8: Modellino macchina

passo-passo, non permetteva alle ruote di assumere posizioni intermedie tra il dritto e il completamente sterzato; inoltre la scheda di controllo era basata su un microcontrollore ATmega48V, che si resettava nel momento in cui mancava l'alimentazione, mentre la batteria fornita era di capacità ridotta e senza alcuna protezione contro la scarica completa indesiderata. Abbiamo quindi deciso di acquistare un modello

(*Figura 4.8)* in scala 1:10 con motore elettrico, equipaggiato di servomotore per regolare lo sterzo, di motore e regolatore elettronico di velocità. Per ricavare lo spazio per i nostri componenti abbiamo installato una rete metallica, collegata al telaio della macchinina, come mostrato in *Figura 4.9,* mentre l'alloggiamento della batteria era già previsto.

Abbiamo scelto questo materiale perché, pur essendo molto leggero e pratico da modellare, è in grado di mantenere la forma anche quando è sottoposto al peso dei componenti, che sono stati disposti in modo da ottimizzare lo spazio a disposizione e da distribuire uniformemente il carico stesso. Per la disposizione abbiamo sfruttato in particolare la parte posteriore, più adatta a sostenere il peso, in quanto non richiede al servomotore uno sforzo eccessivo, garantendo maggiore precisione e tempo di vita. Il posizionamento dei componenti non è identico per entrambi i kart, in quanto le due unità di elaborazione hanno dimensioni molto diverse tra loro; in particolare Odroid è di forma quasi quadrata (48 × 52 mm), mentre Raspberry Pi è rettangolare (85,6 × 53,9 mm). Tutti i pezzi e le strutture presenti sono stati costruiti da noi, utilizzando materiali poco costosi e di facile reperibilità. In *Figura 4.10* sono rappresentati i due kart completi.

Per fornire alle macchinine un aspetto simile a quello di un kart abbiamo

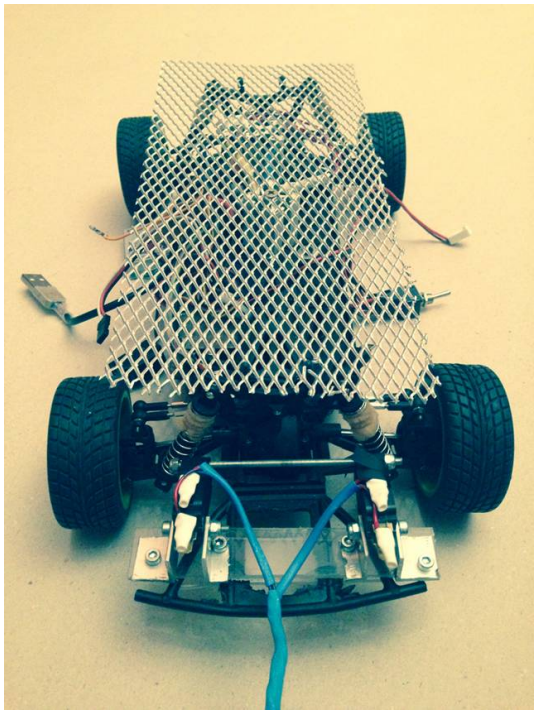


Figura 4.9: Rete metallica

costruito un rivestimento usando della spugna tagliata su misura, successivamente rivestita con del tessuto per darle colore.

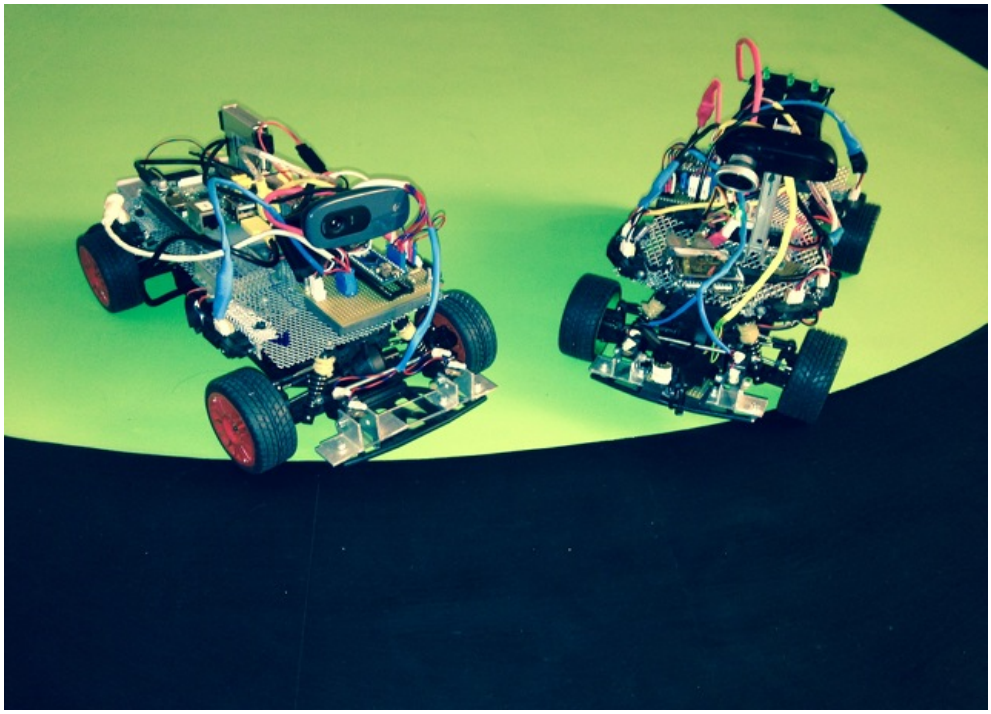


Figura 4.10: Kart

Capitolo 5

Architettura software

Nell'implementazione del software di controllo per i kart, la prima decisione da effettuare è stata se appoggiarci al sistema operativo per robot ROS o creare una soluzione ad-hoc, che andasse incontro alle nostre necessità. La scelta è stata obbligata ed è ricaduta sulla seconda opzione in quanto, avendo inizialmente adottato Raspberry Pi come unità di elaborazione, esso non era in grado di supportare agevolmente ROS, a causa delle sue scarse prestazioni. Abbiamo dunque realizzato interamente il software, utilizzando il linguaggio C++ e sfruttando la libreria OpenCV [26] per l'elaborazione delle immagini.

Al fine di aumentare il più possibile il parallelismo e diminuire la complessità temporale del software, abbiamo creato un thread per la gestione di ogni macrofunzionalità, garantendo così un'elevata modularità. Essi hanno tutti accesso ad una classe implementata come singleton rappresentante lo stato del kart e ciò ha comportato la necessità di prestare particolare attenzione ai problemi di concorrenzialità derivanti dal contesto multithread. Tale oggetto, oltre a realizzare il meccanismo di comunicazione tra i quattro thread presenti, costituisce l'unico punto di accesso per la lettura dei dati sensoriali provenienti da Arduino. Essi vengono inviati da quest'ultimo, tramite la porta seriale, ad una classe C++ apposita, che è anch'essa implementata come singleton e sfrutta, per la loro ricezione, le funzionalità della libreria *LibSerial* [27]. La *Figura 5.1* riporta le interconnessioni appena descritte.

Dal momento che la logica del gioco è implementata su un computer remoto, abbiamo creato un collegamento WiFi tra esso e i kart gareggianti: tale rete ha un'architettura a stella client-server, il cui centro è rappresentato dall'applicazione e i client dai kart. La connessione è stata realizzata tramite

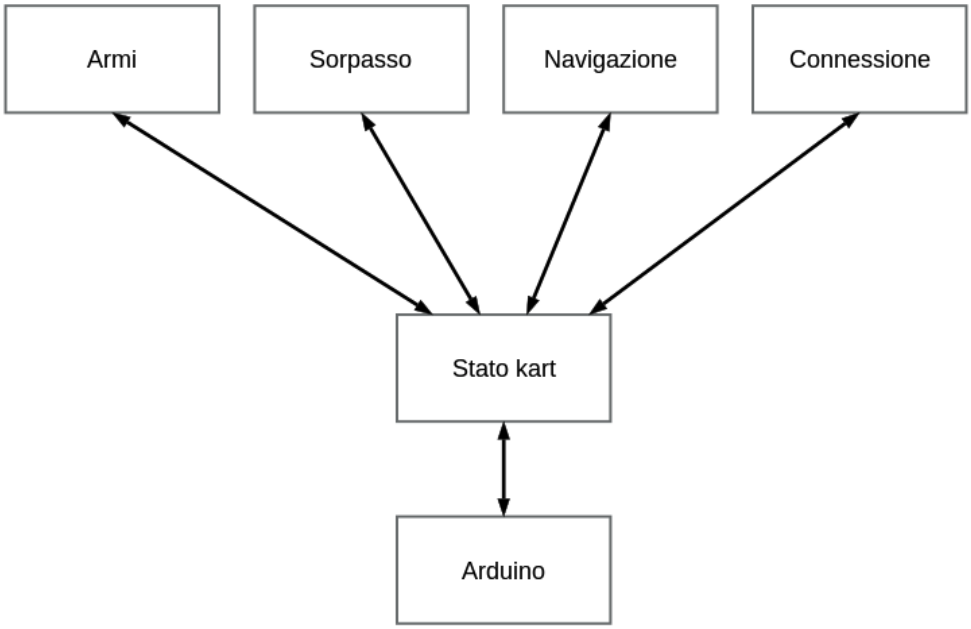


Figura 5.1: Interconnessione logiche software

socket basati su protocollo UDP, per quanto riguarda lo streaming video, e su TCP/IP, per lo scambio di messaggi. In generale, lo schema adottato per la comunicazione, in entrambe le direzioni, prevede l'invio del comando senza attesa di risposta della controparte, in quanto questa non aggiungerebbe nuova informazione, ma aumenterebbe solo il traffico dati sulla rete.

5.1 Visione

La presenza della lente sulla webcam causa una notevole distorsione dell'immagine, che abbiamo dovuto rimuovere, al fine di effettuare in modo adeguato l'elaborazione successiva. In questo processo, vengono presi in cosiderazione solo i contributi della distorsione tangenziale (*Figura 5.2)*, dovuta al fatto che la lente può non essere perfettamente parallela al piano dell'immagine, e della distorsione radiale (*Figura 5.3*), che si manifesta sotto forma di "effetto botte".

Per correggere l'immagine occorre, in primo luogo, individuare la matrice di calibrazione della camera K ed i coefficienti di distorsione C [28] [29] e, successivamente, effettuare una rimappatura dei punti applicando ad essi la trasformazione ottenuta. Quindi, per un punto *(x,y)* dell'immagine origina-

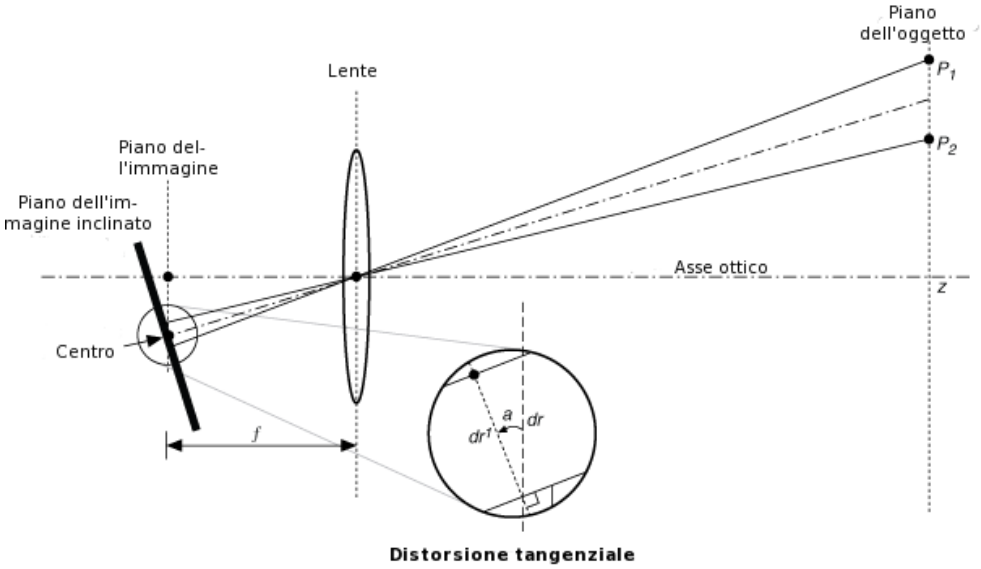


Figura 5.2: Distorsione tangenziale

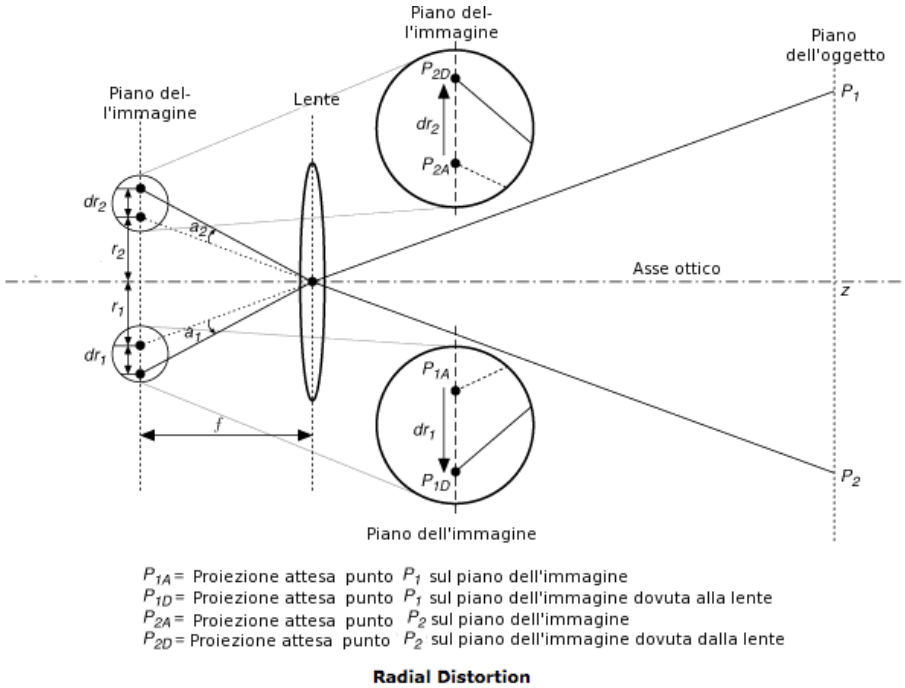


Figura 5.3: Distorsione radiale

le, la sua posizione in quella risultante è $(x_{corretta}, y_{corretta})$, ottenuto secondo l'equazione (5.1), per quel che riguarda la correzione della distorsione radiale, e secondo l'equazione (5.2), per quel che riguarda la correzione di quella tangenziale.

$$\begin{cases} x_{corretta} = x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \\ y_{corretta} = y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \end{cases} \tag{5.1}$$

$$\begin{cases} x_{corretta} = x + [2p_1 xy + p_2(r^2 + 2x^2)] \\ y_{corretta} = y + [p_1(r^2 + 2y^2) + 2p_2 xy] \end{cases} \tag{5.2}$$

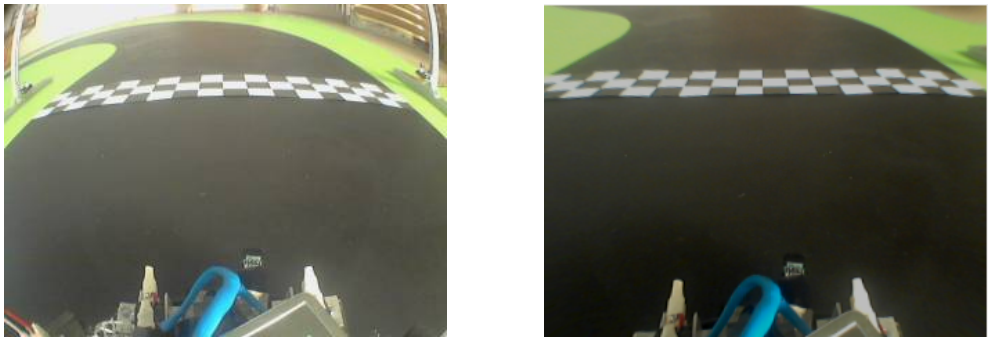
Esse permettono di individuare i cinque parametri k_1, k_2, k_3, p_1, p_2 detti *coefficienti di distorsione,* che in OpenCV vengono considerati come una matrice C di una riga e cinque colonne.

La matrice di calibrazione della camera K, che è generalmente utilizzata per ottenere le coordinate di un punto nel mondo reale date le coordinate nel piano dell'immagine e viceversa, ha la seguente forma:

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

dove f_x e f_y rappresentano la lunghezza focale ed il punto (c_x, c_y) è il centro ottico espresso in pixel.

Il processo che permette di determinare K e C viene detto *calibrazione* [30]. Per ottenerle è necessario acquisire una serie di immagini (nel nostro caso 30), nelle quali siano inquadrati oggetti di forma e dimensioni fisiche note, in modo da poter individuare e risolvere le equazioni geometriche. OpenCV supporta la calibrazione tramite diversi oggetti, tra i quali abbiamo scelto la scacchiera, come conseguenza della maggiore documentazione presente. Il procedimento usato prevede di individuare le coordinate degli angoli dei quadrati interni dell'oggetto nel piano dell'immagine e relazionarle a quelle nel mondo reale, imponendo che ogni punto individuato abbia ascissa pari alla riga della scacchiera in cui si trova ed ordinata pari alla colonna, supposta l'origine del sistema di riferimento nell'angolo in alto a sinistra e sotto l'ipotesi che essa si muova sul piano $z = 0$. In questo scenario, un'unità del sistema di riferimento corrisponde con la lunghezza del lato di un quadrato, che tuttavia è nota a priori nel



(a) Prima (b) Dopo

Figura 5.4: Rimozione della distorsione

mondo. L'impostazione delle equazioni per trovare le matrici è implementata da una funzione di libreria di OpenCV, la quale restituisce le matrici K e C per rimuovere la distorsione introdotta.

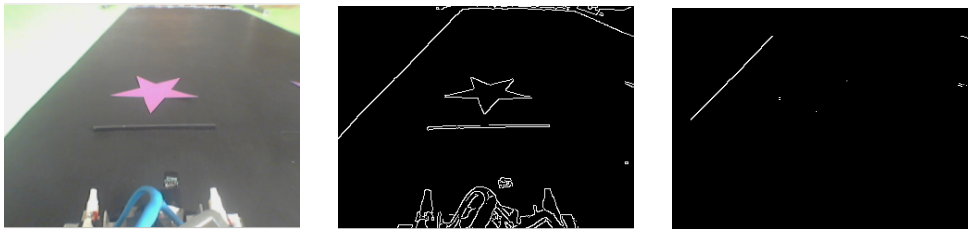
In *Figura 5.4* è possibile osservare l'applicazione della trasformazione calcolata, dall'immagine originale (*5.4a*) al risultato ottenuto (*5.4b*).

5.1.1 Ricerca delle zone transitabili

La ricerca delle zone transitabili, cioè le zone libere della pista su cui il kart può viaggiare, costituisce un compito complesso, dovuto alla decisione di individuare i bordi considerando solo la differenza di colore tra le zone e dall'elevata velocità dello spostamento. In tale situazione, sia il movimento sia le vibrazioni prodotte dalla struttura causano l'acquisizione di immagini non nitide, in cui i contorni possono non essere nettamente definiti.

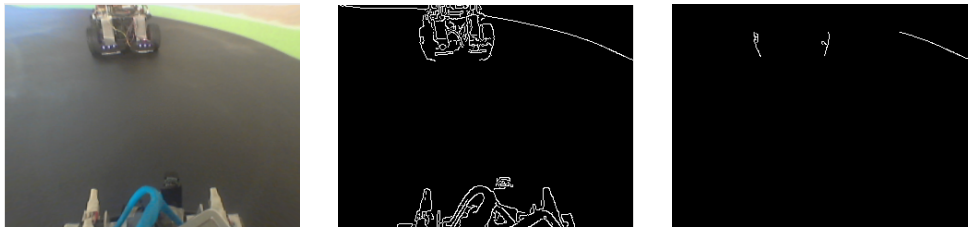
Il nostro approccio è stato quello di ricercare, come prima cosa, la presenza di armi e ostacoli sulla pista e, successivamente, determinare tutti i contorni presenti nell'immagine, identificando tra esse i margini, cioè quelle linee per cui a destra vi è il colore nero e a sinistra il verde, o viceversa.

Il software per il rilevamento dei contorni che abbiamo creato utilizza l'algoritmo di Canny [31], date le sue prestazioni e l'implementazione già esistente in OpenCV, utilizzando soglie determinate sperimentalmente. Esso restituisce una nuova immagine composta solo dai contorni presenti nell'originale, i quali possono essere memorizzati in array di punti, tramite un'apposita funzione di OpenCV. Tale risultato può contenere molte altre linee non utili, come i con-



(a) Immagine originale (b) Algoritmo di Canny (c) Applicazione maschera

Figura 5.5: Rimozione contorni della stella



(a) Immagine originale (b) Algoritmo di Canny (c) Applicazione maschera

Figura 5.6: Rimozione contorni del kart avversario

torni del kart avversario, dell'arma o di oggetti inquadrati non riguardanti il gioco. Per evitare riconoscimenti errati, dovuti all'effetto della prospettiva e a colori simili a quelli ricercati, su tale immagine è applicata una maschera, mediante un'operazione di AND logico, contenente aree rettangolari in posizioni utili ad escludere interamente l'arma individuata e il kart avversario dai contorni rilevati in precedenza.

In *Figura 5.5* è possibile vedere il funzionamento del procedimento nel caso di presenza di una stella: l'immagine originale (*5.5a*), l'applicazione dell'algoritmo di Canny (*5.5b*) e il risultato ottenuto utilizzando la maschera (*5.5c*).

In *Figura 5.6* è possibile osservare lo stesso procedimento eseguito nel caso di presenza di un kart avversario: l'immagine originale (*5.6a*), l'applicazione dell'algoritmo di Canny (*5.6b*) e il risultato ottenuto utilizzando la maschera (*5.6c*).

Tra i rimanenti, abbiamo identificato come bordo destro o sinistro, a seconda dei casi, quelli che contengono almeno tre punti per cui valgono le condizioni in *Tabella 5.1,* verificate controllando per ciascuno un numero di pixel pari a 40, se esistono, altrimenti il numero massimo possibile prima di raggiungere

Tipo bordo	***Condizioni***
Sinistro	• a destra del punto almeno 1/3 dei pixel controllati sono neri e a sinistra almeno 1/3 sono verdi; • a destra del punto almeno 1/3 dei pixel controllati sono neri, a sinistra il numero dei pixel neri è minore di 1/6 e a destra il numero di pixel verdi è minore di 1/6; • a sinistra del punto almeno 1/3 dei pixel controllati sono verdi, a destra il numero dei pixel verdi è minore di 1/6 e a sinistra il numero dei pixel neri è minore di 1/6.
Destro	• a sinistra del punto almeno 1/3 dei pixel controllati sono neri e a destra almeno 1/3 sono verdi; • a sinistra del punto almeno 1/3 dei pixel controllati sono neri, a destra il numero dei pixel neri è minore di 1/6 e a sinistra il numero di pixel verdi è minore di 1/6; • a destra del punto almeno 1/3 dei pixel controllati sono verdi, a sinistra il numero dei pixel verdi è minore di 1/6 e a destra il numero dei pixel neri è minore di 1/6.

Tabella 5.1: Identificazione bordo

la fine dell'immagine, sia a destra sia a sinistra di esso. La ricerca termina quando sono stati analizzati tutti i contorni individuati.

5.1.2 Rilevazione indicatore delle armi

Un indicatore delle armi viene riconosciuto basandosi sia sul colore sia sulla forma. Per identificare il primo abbiamo convertito l'immagine nello spazio HSV, filtrandola poi con l'utilizzo di specifiche soglie, determinate sperimentalmente al fine di individuare il colore viola con qualsiasi illuminazione ambientale. Questa operazione restituisce un'immagine composta da aree bianche su sfondo nero, che rappresentano le zone in cui è presente il colore ricercato. Da essa abbiamo estratto i contorni tramite l'apposita funzione di OpenCV e, dopo aver scartato quelli con area inferiore a 100 pixel, abbiamo determinato per ciascuno dei rimanenti il suo *involucro convesso*, cioè il più piccolo insieme

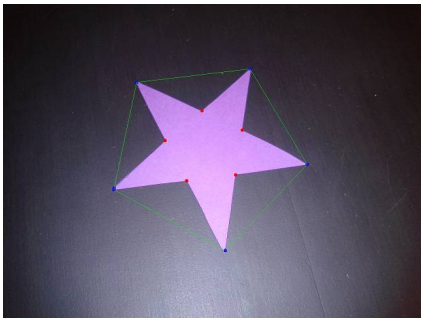


Figura 5.7: Elaborazione arma

che contiene tutti i suoi punti, come mostrato in *Figura 5.7*. Tale involucro è necessario per ottenere i cosiddetti *difetti di convessità*, cioè strutture composte da un punto iniziale, che indica una punta della stella, un punto finale, corrispondente alla punta successiva, un punto di profondità, costituito dal vertice interno compreso tra esse e un valore di profondità, calcolato come la distanza tra quest'ultimo e il segmento che congiunge punto iniziale e punto finale. Abbiamo identificato come arma tutte le aree viola per cui esistono dai 3 ai 6 difetti di convessità, per permetterne il riconoscimento anche in caso di illuminazione non ottimale o di non completa inquadratura nell'immagine. Se è presente più di un'arma, viene considerata la più vicina, cioè quella per cui è necessaria una variazione minore dalla direzione rispetto a quella attuale.

5.1.3 Riconoscimento del kart avversario

Il riconoscimento del kart avversario avviene tramite la wiiCam, che rileva le coordinate di massimo quattro sorgenti nell'infrarosso, espresse in un sistema di riferimento, la cui origine è posta nell'angolo in alto a sinistra e la cui risoluzione è di 1024 $\times$ 768 pixel. Tali coordinate non possono essere usate direttamente per tracciare l'ostacolo sull'immagine, in quanto la webcam e la wiiCam sono installate ad altezze diverse sul kart, seppur entrambe in posizione centrale. La trasformazione che lega i punti rilevati dalle due camere è un'omografia, identificata da una matrice H, di dimensione 3 $\times$ 3, come indicato nella (5.3).

$$H = \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & 1 \end{bmatrix} \tag{5.3}$$

Dato un punto P_{wiiCam} identificato dalla wiiCam, esso verrà trasformato nel punto P_{webcam}, secondo l'equazione (5.4).

$$P_{webcam} = H \cdot P_{wiiCam} \tag{5.4}$$

Per identificare gli elementi della matrice H abbiamo sfruttato un'apposita funzione di OpenCV, la quale, fornendo la corrispondenza tra le coordinate di quattro punti individuati manualmente, risolve un sistema di otto equazioni linearmente indipendenti e restituisce le incognite h_i.

5.1.4 Elaborazione dell'immagine

Al fine di ridurre l'elaborazione e incrementare le prestazioni del software, abbiamo deciso di considerare solo una fascia dell'immagine che, data l'altezza a cui è montata la webcam, comprende i pixel dall'ordinata 30 alla 120. Ciò diminuisce, inoltre, la probabilità di inquadrare zone esterne alla pista, assicurando, al tempo stesso, di individuare correttamente tutte quelle transitabili. Per non analizzare ogni pixel contenuto in tale area, abbiamo deciso di raggrupparli in regioni d'interesse dette *ROI (Region Of Interest),* ognuna delle quali ha una dimensione di 16 $\times$ 6 pixel, determinata sulla base di un compromesso tra la riduzione della stessa, che avrebbe permesso di ottenere un risultato più preciso, e il suo incremento, che avrebbe ridotto il tempo di elaborazione, favorendo un comportamento più reattivo del kart. Poiché siamo interessati unicamente allo stato di ciascuna ROI, abbiamo deciso di memorizzarlo in un array bidimensionale di interi. Il contenuto dell'elemento corrispondente può assumere uno dei seguenti valori:

- ***area libera***, se costituisce uno spazio in cui il kart può transitare, identificato sull'immagine dalle celle verdi;
- ***fuori pista***, se rappresenta un'area fuori dai contorni della pista, identificato sull'immagine dalle celle marroni;
- ***ostacolo***, se è occupata da un kart avversario, identificato sull'immagine dalle celle rosse;
- ***arma***, se contiene il centro di un'arma, identificato sull'immagine dalle celle azzurre.

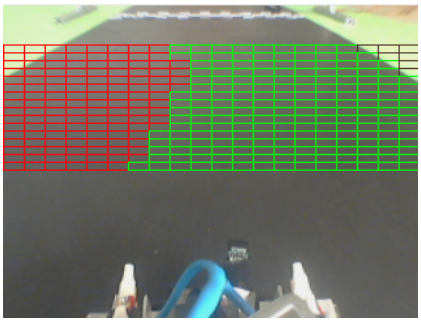
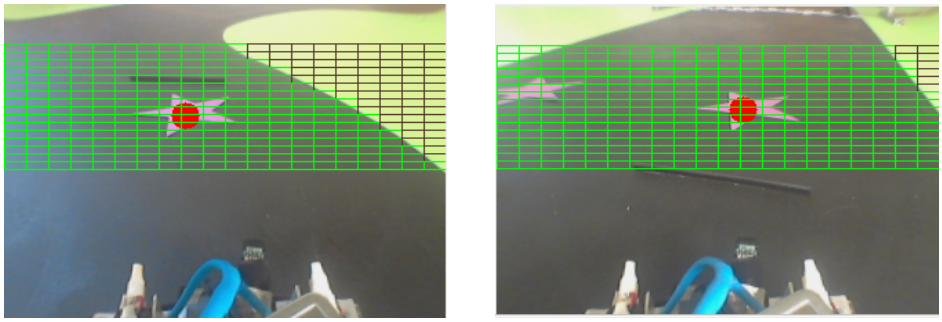


Figura 5.8: Ostacolo laterale a sinistra

La matrice viene inizializzata ponendo tutti gli elementi al valore indicante il fuori pista e successivamente modificata usando le informazioni ottenute sui bordi, sulle armi e sull'avversario. Per ogni riga vengono marcate come aree libere le ROI che vanno dal bordo sinistro al destro, nel caso entrambi siano stati individuati, dal bordo sinistro a fine immagine, se è stato individuato solo quello, o dall'inizio dell'immagine a quello destro, nel caso opposto. Nel caso in cui non venga trovato alcun bordo, nessuna ROI viene marcata come area libera. Se vengono riconosciuti indicatori a forma di stella, la ROI corrispondente al centro di essi viene indicata come arma e, se viene individuato un avversario, alle ROI corrispondenti alla sua posizione viene assegnato il valore di ostacolo. Inoltre, quando i sensori laterali percepiscono la presenza di un kart, l'area libera di ogni riga viene ristretta da quel lato. Ciò avviene marcando un numero x di ROI come ostacolo, partendo dal bordo corrispondente al lato da cui è stato rilevato e muovendosi verso l'interno della pista, con x pari alla differenza tra il centro dell'area libera e metà della larghezza che un kart occuperebbe su quella riga, espressa in ROI. Un esempio di tale situazione, in caso di ostacolo sul lato sinistro, è visibile in *Figura 5.8*.

5.2 Scelta della prossima azione

La decisione della prossima azione da intraprendere costituisce un aspetto di notevole importanza, al fine di eseguire movimenti intelligenti e garantire la massima fluidità, evitando bruschi cambi di direzione. Un elemento essenziale, ad esempio, è costituito dalla modulazione della velocità a seconda del tipo di



(a) Arma singola (b) Arma doppia

Figura 5.9: Goal con arma

tracciato che si sta per intraprendere, proprio come avviene anche nella realtà quando una persona è alla guida di un'automobile: è più elevata in situazioni favorevoli, come in un rettilineo o in una curva dolce, e ridotta quando invece ci si avvicina ad una curva stretta o la situazione è particolarmente pericolosa.

Il passo successivo alla rilevazione dei bordi sull'immagine è quello dell'identificazione della tipologia di tracciato: rettilineo, curva a destra o curva a sinistra. Per fare ciò, viene osservato il bordo contenente il numero maggiore di punti, il che fornisce anche una stima approssimata della posizione del kart in pista. Osservando l'andamento della retta $y = mx + q$, passante per il primo e l'ultimo punto del bordo analizzato, è possibile classificarlo come curva a destra, se tutti i suoi punti giacciono nell'iperpiano $y < mx + q$, come curva a sinistra, se tutti i suoi punti giacciono nell'iperpiano $y > mx + q$, o come rettilineo, se tutti i suoi punti appartengono alla retta.

La direzione da attribuire al kart per il movimento successivo è quella che lo porta a orientarsi verso un punto identificato sull'immagine, detto *goal*, calcolato sfruttando le informazioni contenute nella matrice delle ROI e la tipologia del tracciato. Nel caso in cui vengano riconosciute una o più armi, esso viene posizionato in corrispondenza della ROI che rappresenta quella più vicina, come visibile in *Figura 5.9*, con un'arma singola (*5.9a*) e con due armi (*5.9b*).

Se viene rilevato il kart avversario, situazione che ha priorità maggiore rispetto alla precedente, esso è posizionato in un'area libera sufficientemente larga da permettere il passaggio in completa sicurezza. La ricerca di quest'ultima è effettuata nella zona più vicina al kart rappresentante l'avversario,

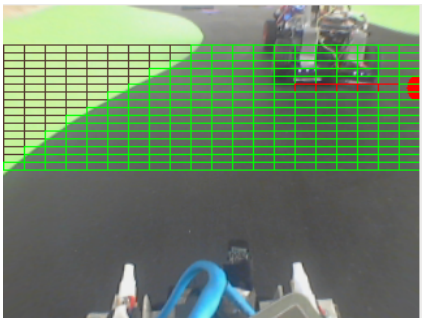


Figura 5.10: Area indefinita sul lato destro

cioè la riga di ROI più bassa nell'immagine, avente celle marcate come ostacolo. La larghezza di un'area è misurata in pixel sull'immagine e convertita in centimetri usando l'equazione (5.5), dove il coefficiente c_x, calcolato sperimentalmente, indica il rapporto tra la larghezza di una ROI in pixel sulla riga x e la sua dimensione in centimetri nel mondo reale.

$$l_{cm} = c_x \cdot l_{pixel} \tag{5.5}$$

L'analisi è effettuata identificando le sequenze di ROI consecutive marcate come libere e stabilendo, per ognuna, se è ampia a sufficienza per il passaggio del kart e, in caso ciò sia vero, se è la più larga tra quelle finora scoperte. Il goal viene infine posizionato in corrispondenza della ROI centrale dell'area maggiore individuata. In situazioni particolari, può accadere che una o entrambe le aree libere a lato dell'avversario non siano completamente inquadrate nell'immagine. Un esempio del primo caso è mostrato in *Figura 5.10.* In essa si può notare come l'area libera più larga presente nell'immagine sia a sinistra, mentre, nella realtà, la soluzione migliore sarebbe effettuare il sorpasso a destra. Ciò si verifica perché la porzione più grande della pista non rientra nell'inquadratura. Per risolvere questo problema e permettere al kart di esplorare tale zona, abbiamo assegnato ad un'area libera non delimitata, cioè tale per cui la prima o l'ultima ROI coincida con il bordo dell'immagine, una larghezza idealmente infinita, che quindi le permetterà di essere sicuramente scelta.

Qualora in un'immagine entrambe le aree libere in corrispondenza del punto più vicino dell'avversario non siano interamente inquadrate, come in *Figura 5.11,* il kart non è in grado di identificare con certezza la prossima azione da

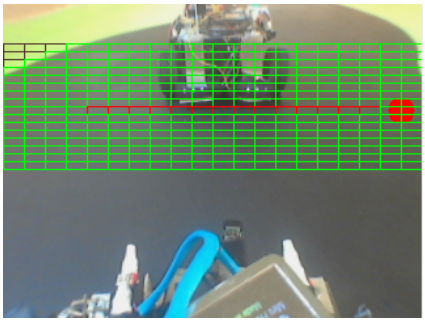
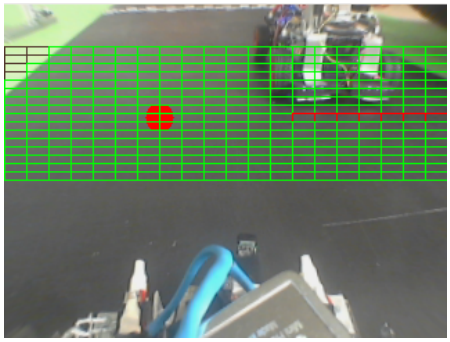
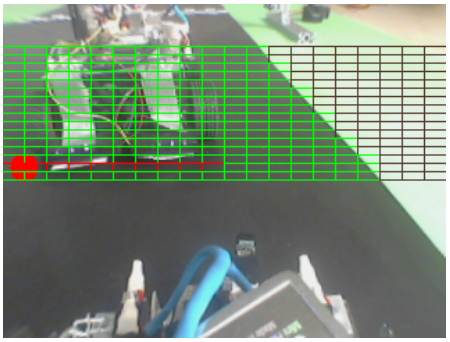


Figura 5.11: Area indefinita su entrambi i lati



(a) Spazio sufficiente



(b) Spazio insufficiente

Figura 5.12: Ostacolo

eseguire, in quanto non è possibile conoscere la sua posizione all'interno del tracciato. Quando ciò si verifica, abbiamo ritenuto opportuno sfruttare la più recente informazione nota sui bordi, nell'immagine corrente o in quelle precedenti: supponendo che l'ultimo bordo visto sia il più vicino alla posizione attuale del kart, il goal viene posto nell'area libera dall'altro lato dell'avversario, dove presumibilmente si trova lo spazio più grande.

Altra situazione particolare è rappresentata da immagini in cui è presente spazio solo ad un lato dell'avversario, in quanto esso occupa totalmente la prima o la seconda porzione dell'inquadratura. Se ciò si verifica e l'unica area libera trovata non permette il passaggio, il goal viene posizionato nel punto più estremo dell'immagine, dal lato dove è presente il kart, supponendo che la porzione maggiore della pista si trovi da quella parte.

In *Figura 5.12,* è possibile osservare tale situazione in due diversi casi: nel primo, lo spazio delimitato visto è sufficiente al passaggio (*5.12a*), nel secondo

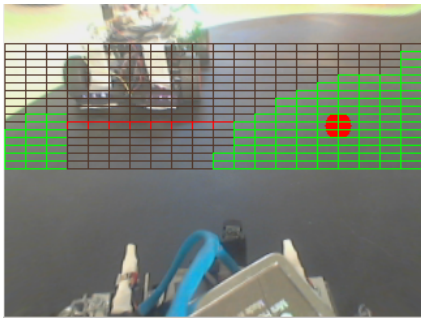


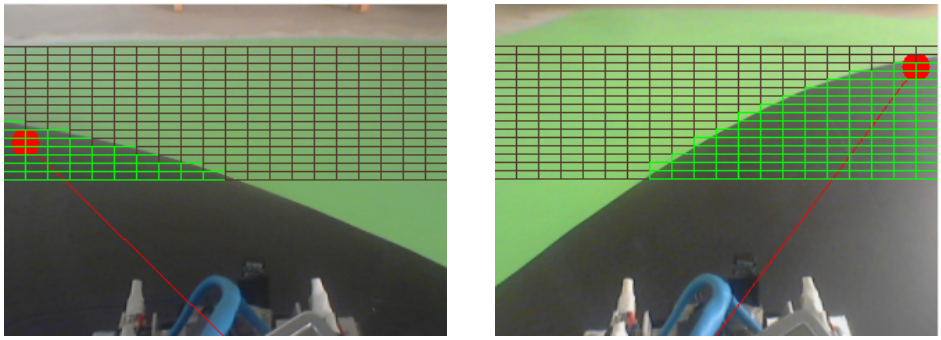
Figura 5.13: Approssimazione in assenza di bordi

invece no (*5.12b*).

In qualunque caso di scelta errata della direzione da percorrere, il kart esegue il movimento selezionato, per poi correggerlo non appena riceve informazioni utili dalle immagini successive. Vi sono poi situazioni in cui il kart avversario è posizionato in modo da impedire il riconoscimento dei bordi, come in *Figura 5.13*. In tal caso, l'unica informazione utilizzabile per impedire la fuoriuscita dal tracciato e l'urto con esso è quella relativa alla posizione del colore nero rilevato. Le ROI vengono marcate con area libera se contengono un numero di pixel neri pari a più di metà di quelli totali. Seppur ciò costituisca un'approssimazione, permette comunque al kart di effettuare un'azione non totalmente casuale, cosa altrimenti non possibile.

In tutti i casi in cui la direzione da seguire non è identificata con certezza, il kart attua un comportamento conservativo, rallentando fino a raggiungere la velocità minima. Nel caso in cui vengano individuati sia l'avversario sia l'arma, abbiamo ritenuto opportuno dare la precedenza all'evitamento della collisione.

Nelle situazioni in cui nell'immagine non sono identificati né il kart avversario né le armi, distinguiamo tre diversi casi. Il primo è quello della presenza di una curva particolarmente stretta e si verifica, ad esempio, quando il kart si sta avvicinando eccessivamente ad un bordo, sia nell'esecuzione di una curva sia quando esso è direzionato in modo da avere un lato di fronte a sè, come in *Figura 5.14a*. Ciò è individuato contando il numero di righe visibili esterne al tracciato, partendo dalla parte alta dell'immagine, cioè dalla zona più lontana dal kart. Perché una riga sia considerata tale, deve avere almeno il 75% delle sue ROI marcate come fuori pista. Il numero ottimale per assicurare fluidità al movimento, in relazione alla velocità, è che il totale di esse sia maggiore di 5. Abbiamo determinato tale valore sperimentalmente: un numero più alto



(a) Curva stretta (b) Curva intermedia

Figura 5.14: Tipi di curva

causerebbe un eccessivo avvicinamento al bordo, il che richiederebbe poi una brusca sterzata, mentre uno più basso lo farebbe girare troppo presto rispetto alla posizione della curva. In tale situazione, il goal viene posizionato sulla prima colonna (in caso di curva a sinistra) o sull'ultima (in caso di curva a destra) e spostato verso il basso, partendo dalla prima riga in alto in cui è presente area libera, proporzionalmente al numero di righe fuori pista rilevate.

Il secondo caso che può presentarsi è costituito dalla presenza di una curva vicina, ma per cui la situazione non è ancora estrema come la precedente. Ciò è individuato da un numero di righe esterne alla pista comprese tra 2 e 5. Quando tale condizione si verifica il goal viene posizionato inizialmente nel centro dell'area libera più stretta visibile nell'immagine, trovata controllando tutte le righe, e spostato a sinistra o destra, a seconda del tipo di curva, in modo proporzionale al numero di righe fuori dal tracciato visibili. Tale situazione è osservabile in *Figura 5.14b*.

In tutti i casi rimanenti, cioè rettilinei e curve dolci o ancora lontane, abbiamo adottato un metodo basato su considerazioni geometriche per stabilire il goal: viene costruito un triangolo in cui due vertici sono costituiti dal centro della prima e dell'ultima ROI libere, nella riga più bassa avente elementi liberi, mentre il terzo è posto a metà della sequenza di ROI consecutive più stretta, trovata controllando ciascuna riga. Il goal viene posto nel centro della ROI contenente la mediana del triangolo così definito. Tale metodo permette al kart di avvicinarsi al bordo interno della pista nelle curve e di rimanere invece al centro del tracciato quando sta percorrendo un rettilineo. Nella *Figura 5.15* è mostrato il posizionamento del goal mediante il triangolo.

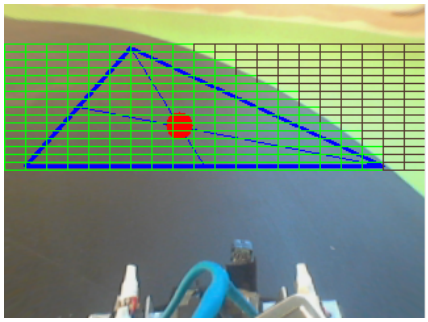


Figura 5.15: Goal con triangolo

5.3 Esecuzione del movimento

Il movimento eseguito dal kart avviene in direzione della posizione del goal, considerando come punto iniziale quello avente come ordinata l'altezza dell'immagine, essendo l'origine del sistema di riferimento posta in alto a sinistra, e ascissa pari a metà della sua larghezza. L'inclinazione delle ruote è ricavata misurando l'angolo compreso tra l'asse delle ascisse e la retta passante per il goal e il punto iniziale. Anche la velocità attribuita al kart dipende da tale valore: esso viene rallentato se l'angolo si avvicina agli estremi (0° e 180°) e accelerato in prossimità dei 90°.

Per descrivere il nostro mondo tridimensionale siamo abituati ad usare la geometria euclidea, dove ogni oggetto ha delle dimensioni, due linee intersecanti formano un certo angolo e le linee parallele non si incontrano mai. Quando consideriamo un'immagine, tale geometria non è più sufficiente, in quanto non vengono preservate le dimensioni e gli angoli degli oggetti reali e le linee parallele possono intersecarsi. Per questo motivo, non è possibile effettuare alcuna misura direttamente sull'immagine, senza prima rimuovere l'effetto della proiezione. Per fare ciò, è necessario identificare una matrice che, applicata all'immagine, la restituisca rettificata. Il procedimento che abbiamo utilizzato è quello descritto in [32]. La scena su cui ci siamo basati era costituita da un foglio A4 posto sul pavimento, il quale può essere considerato ad una quota pari a 0 rispetto al sistema di riferimento con origine nella camera. Dal momento che i rettangoli nel mondo reale sono proiettati nel piano dell'immagine sotto forma di quadrangoli, l'obiettivo è stato trovare la matrice che effettuasse tale trasformazione, mantenendo le proporzioni tra i lati. Una volta ottenuta, per non appesantire eccessivamente l'elaborazione, abbiamo scelto di applicarla so-

Condizioni	***Incremento***
RETTILINEO e | 90-angolo|<15	*+1*
RETTILINEO e |90-angolo|>15	*0*
CURVA e |90-angolo|<25	*+1*
CURVA e |90-angolo|>25	0

Tabella 5.2: Incremento velocità

lo al goal e al punto iniziale, così che l'angolo formato da tale retta coincidesse con quello nel mondo reale.

Poiché non vi è una corrispondenza diretta tra gli angoli calcolati sull'immagine, che sono crescenti in senso antiorario, e quelli del servomotore, crescenti in senso orario, per posizionare le ruote alla corretta inclinazione è stato necessario inviare ad Arduino il supplementare del valore ottenuto dall'immagine. Idealmente la posizione delle ruote diritte dovrebbe, quindi, corrispondere ad un angolo di 90° assegnato al servomotore, a metà tra gli estremi rappresentati da 0° e 180° (in realtà ristretti a 40° e 140° per evitare di sottoporlo a sforzi eccessivi non necessari), cosa che nella realtà non sempre si verifica e varia da un componente ad un altro. Per rendere la nostra soluzione flessibile, comunque, nell'algoritmo viene calcolata l'angolazione ideale da fornire al servomotore. Essa viene poi traslata da Arduino prima dell'invio del comando, in modo da far coincidere il centro ideale con quello reale del servomotore, dichiarato nel programma sviluppato per Arduino (sketch).

Il valore della velocità è invece generato da un controllore di tipo fuzzy, in cui le variabili d'ingresso sono costituite dalla tipologia di tracciato e dalla differenza tra l'angolo di posizionamento calcolato per le ruote e 90°, gli insiemi fuzzy sono identificati dalle condizioni descritte in *Tabella 5.2* e la variabile d'uscita è l'incremento da applicare alla velocità minima del kart. Nel caso in cui il valore relativo al sensore di colore letto da Arduino indichi che quest'ultimo si trova al di fuori del tracciato, tale velocità verrà sempre posta al minimo. Ciò, ovviamente, non avviene solo per il kart autonomo, ma anche per quello controllato, al fine di dissuadere il giocatore dal tagliare parti della pista.

Ruote kart	***Valore accelerometro***	***Angolo servomotore***
Tutto sinistra	*100*	*40°*
Dritte	*125*	*90°*
Tutto destra	*150*	*140°*

Tabella 5.3: Corrispondenza angoli

5.4 Gestione della gara per il kart controllato

Il controllo del kart telecomandato avviene mediante Wii Remote, connesso via bluetooth all'unità di elaborazione, sfruttando i metodi della libreria *bluez* [33]. La gestione della gara prevede, in questo caso, l'implementazione di tre sottofunzionalità: la ricezione dei comandi, la loro interpretazione e l'invio ad Arduino. I messaggi scambiati vengono gestiti tramite la libreria *libcwiimote* [34], che prevede l'esistenza di una struttura rappresentante l'oggetto Wii Remote, i cui campi contengono informazioni sul suo stato. Per le necessità richieste dalla gara, siamo interessati ad identificare quattro tipi di eventi:

- pressione del tasto "1": se schiacciato, provoca l'incremento della velocità del kart fino al suo valore massimo (circa 2 m/s) e, se rilasciato, il suo decremento fino al valore minimo;
- pressione del tasto "2": ferma istantaneamente il kart;
- pressione del tasto "B": permette di utilizzare l'arma a disposizione, mediante l'invocazione di un'apposita funzione del gestore delle armi;
- inclinazione del Wii Remote rispetto alla posizione orizzontale: si riflette sull'orientamento delle ruote.

Data la struttura della libreria, l'evento "bottone premuto" è identificato tramite il test sulla corrispondente variabile booleana, costantemente aggiornata. L'inclinazione del Wii Remote rispetto all'orizzontale viene rilevata, invece, sfruttando le informazioni provenienti dall'accelerometro presente in esso, prestando attenzione solo ai valori riguardanti l'asse y del sistema di riferimento rappresentato in *Figura 5.16*.

Gli angoli così ottenuti non corrispondono direttamente, però, a quelli del servomotore. In *Tabella 5.3* è riportata la relazione esistente tra essi.

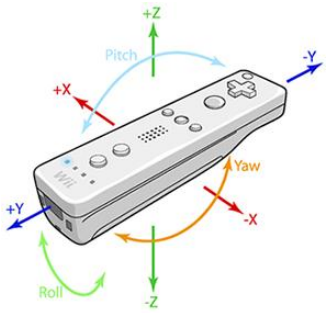


Figura 5.16: Sistema di riferimento wiimote

L'angolo da assegnare al servomotore è quindi calcolato secondo l'equazione (5.6), considerando, analogamente al caso del kart autonomo, come diritta la posizione in cui il servo riceve un'angolazione di 90°.

$$angolo_{servomotore} = (angolo_{wiimote} - 100) * 100/50 + 40 \tag{5.6}$$

Nel caso in cui il giocatore fuoriesca dalla pista, il Wii Remote è utilizzato anche per restituirgli un feedback mediante la vibrazione.

5.5 Armi

Dal punto di vista del software possiamo individuare tre aspetti riguardanti le armi: il riconoscimento da parte del kart di averla acquisita, la selezione della tipologia e l'utilizzo, comprendente l'applicazione dei suoi effetti. A causa dell'elevata frequenza con cui queste operazioni devono essere effettuate, abbiamo deciso di creare un thread che implementi queste funzionalità.

Il riconoscimento dell'acquisizione di un'arma è realizzato da Arduino, il quale, al momento della chiusura di uno degli interruttori, pone una variabile booleana locale al valore vero. Quest'ultima è costantemente controllata dal thread, per rilevare il verificarsi dell'evento e attuare le operazioni necessarie. Per rendere visibile al giocatore l'acquisizione dell'arma, Arduino accende una serie di led verdi posti sul kart, i quali rimangono accesi per tutto il tempo in cui ne è in possesso. Al momento della ricezione della notifica *WEAPON_EXPLOITED* (*Sezione* 5.8), Arduino pone al valore falso tale variabile booleana e spegne i led.

Arma	***Valore***
Immunità	*0*
Velocità	*1*
Fulmine	*2*
Vibrazione	3
Stop	4

Tabella 5.4: Discretizzazione armi

Per implementare la scelta pseudo-casuale della nuova arma vi erano due opzioni possibili: una scelta centralizzata, cioè effettuata dall'applicazione alla ricezione della notifica di acquisizione da parte del kart, che avrebbe implicato una risposta contenente la tipologia selezionata, o una scelta decentralizzata, in cui è il kart stesso a effettuarla in modo automatico, inviando all'applicazione una semplice notifica di tipo informativo. Abbiamo scelto di utilizzare la seconda opzione, in quanto è di più semplice realizzazione e si adatta meglio al sistema per lo scambio di informazioni in rete. Se la scelta avvenisse in modo totalmente casuale, un kart potrebbe risultare ripetutamente avvantaggiato o svantaggiato. Abbiamo quindi fatto sì che essa sia vincolata alla posizione occupata, di cui ogni kart è a conoscenza: quello in prima posizione può acquisire unicamente l'immunità, mentre il kart in seconda posizione tutte le altre armi, con identica probabilità. In realtà, la selezione delle armi è stata implementata in modo da essere già espandibile ad un numero di kart superiore a due: in questo caso, i giocatori nelle prime posizioni hanno una probabilità maggiore di ottenere l'immunità e minore di ottenere le altre armi, mentre avviene il contrario nelle posizioni più arretrate. Nello specifico, il kart al primo posto ha probabilità pari a 1 di ricevere l'immunità, probabilità che decresce linearmente con l'aumentare della posizione, fino a diventare 0 per l'ultima. La probabilità restante viene suddivisa tra le altre quattro armi, che continuano ad essere equiprobabili tra loro. Matematicamente, possiamo definire la variabile aleatoria X_n come la scelta di una nuova arma essendo in posizione n. Il valore assunto da essa è discreto, in quanto il dominio è rappresentato dalle armi, ad ognuna delle quali corrisponde un intero, secondo la *Tabella 5.4*. Ad inizio gara ogni kart calcola la distribuzione di probabilità di X_n, per ogni posizione n disponibile. Da questa vengono calcolate le funzioni

di ripartizione F_n, i cui valori sono poi memorizzati in una tabella che ha come righe le posizioni possibili e come colonne le armi esistenti. L'elemento *(i,j)* rappresenta, quindi, il valore della funzione di ripartizione in corrispondenza della variabile aleatoria X_i e di j. La scelta della nuova arma avviene tramite un esperimento di estrazione di un numero casuale x compreso tra 0 e 1 e la successiva selezione dell'arma a per cui vale l'equazione (5.8)

$$P(X_n \leq a) < x < P(X_n < a + 1) \tag{5.7}$$

$$F_n(a) < x < F_n(a + 1) \tag{5.8}$$

La nuova arma acquisita viene memorizzata in una variabile nella classe rappresentante lo stato del kart.

Il momento in cui l'arma viene utilizzata è scelto dall'utente tramite Wii Remote nel kart controllato, mentre nel kart autonomo l'arma è utilizzata immediatamente appena acquisita, tranne nel caso del razzo, sfruttato nel primo rettilineo disponibile. Gli effetti delle armi vengono applicati dai kart stessi e questo comporta la necessità di comunicare all'avversario l'azione di un'arma svantaggiosa su di lui. Per evitare che ciò avvenga in modo diretto, il kart che vuole utilizzarla invia una notifica all'applicazione, la quale poi si occupa di informare l'altro. Ovviamente, per quanto riguarda l'immunità e il razzo, che agiscono solo sul kart stesso, il messaggio inviato costituisce solo una notifica dell'avvenuto utilizzo, senza necessità di intervento da parte dell'applicazione. Nel corso della gara è molto probabile che capitino situazioni in cui, mentre un'arma non ha ancora terminato il suo effetto sul kart, ve ne sia una nuova che inizia ad agire. In tal caso l'effetto globale è ottenuto come la somma degli effetti prodotti dalle singole armi.

Dal punto di vista software gli effetti delle armi sono stati implementati come classi C++, ognuna delle quali estende una classe astratta, contenente le informazioni relative al tipo di arma, alla durata dell'azione e alle modifiche da applicare alla velocità. Ogni volta che il kart utilizza un'arma vantaggiosa o che viene ricevuto un messaggio indicante che è stato colpito dall'avversario, il vettore che contiene le armi agenti in quel momento viene modificato di conseguenza. Per rendere poi gli effetti visibili, prima di inviare la velocità ad Arduino, il software invoca una procedura che adegua il valore di essa sulla base delle armi agenti e produce un suono caratteristico.

5.6 Sorpasso

Il riconoscimento del sorpasso consiste nell'interpretazione dei valori relativi ai sensori laterali, la cui lettura avviene ad opera di Arduino e che sono memorizzati all'interno della classe rappresentante lo stato del kart. Poiché tale operazione deve essere effettuata con frequenza elevata, abbiamo creato un thread che implementi questa funzionalità.

Ogni sensore di prossimità restituisce un valore compreso tra 0 e 600 alla scheda di Arduino, inversamente proporzionale alla distanza, idealmente uguale 0 quando non c'è alcun ostacolo e tendente a 600 più l'oggetto rilevato è vicino. Nella realtà, quando è presente un ostacolo il valore restituito può essere considerato molto simile a quello ideale, mentre quando non è presente ci possono essere delle imprecisioni, dovute sia a condizioni sfavorevoli nell'ambiente sia alla scarsa qualità dei sensori. Per tali motivazoni, abbiamo identificato l'assenza del kart avversario davanti ad un sensore quando esso restituisce un valore minore di una soglia, determinata sperimentalmente al valore 50. Tale operazione viene effettuata da Arduino e le informazioni relative allo stato di ogni sensore vengono salvate in una variabile, la cui struttura è descritta in *Figura 5.17*. Ogni bit di essa è relativo ad un sensore e viene settato a 1 se è presente un ostacolo, a 0 altrimenti. Tale variabile viene inviata all'unità di elaborazione a fronte di un'opportuna richiesta, come riportato nella *Sezione 5.8*.

5.7 Connessione

Ogni kart è stato configurato assegnandogli un indirizzo IP WiFi statico appartenente alla rete 192.168.1.0/24 e l'indirizzo IP 10.0.0.1 per la scheda di rete ethernet, uguale per tutti i kart.

Date le funzionalità richieste, il kart deve mantenere attive due connessioni contemporaneamente: una per inviare lo streaming video all'applicazione, basata su protocollo UDP, e un'altra per lo scambio dei messaggi riguardanti le dinamiche del gioco, basata su TCP/IP. Entrambe sono implementate sfruttando il meccanismo dei socket e utilizzando la porta 7654.

Lo streaming video prevede l'invio di immagini, ciascuna preceduta da un pacchetto rappresentante il nome utente del kart, così che l'applicazione possa sia riconoscere da chi proviene sia sincronizzarsi sulla ricezione. La struttu-

ra dei messaggi inviati è descritta nel *Capitolo 6*, i quali, dal punto di vista software, sono stringhe terminanti con il carattere '\n'. I messaggi vengono ricevuti da un thread, sempre attivo ed in ascolto sulla porta desiderata, avente accesso ai campi della classe rappresentante lo stato del kart e vengono decodificati e interpretati mediante l'utilizzo di espressioni regolari.

5.8 Arduino

Il funzionamento dello sketch software caricato sulla scheda di Arduino prevede che esso rimanga in ascolto sulla porta seriale per ricevere ed elaborare le seguenti richieste ricevute dall'unità di elaborazione, ognuna identificata da un carattere della dimensione di un byte:

- *VEL_STEERING_CMD*: Arduino attende l'invio di due valori, il primo dei quali rappresentante la nuova velocità e il secondo il nuovo angolo da assegnare al servomotore che controlla lo sterzo;
- *READ_STATUS:* Arduino risponde con un byte indicante lo stato di tutti i sensori di prossimità e della variabile che identifica l'avvenuta acquisizione di un'arma;
- *READ_COLOR*: Arduino legge e invia il valore rilevato dal sensore di colore;
- *WEAPON_EXPLOITED*: Arduino riceve la notifica che l'arma è stata utilizzata, spegne i led e setta la variabile booleana locale di possesso dell'arma a falso;
- *OPPONENT_KART*: Arduino invia le coordinate *(x,y)* delle sorgenti infrarosso rilevate dalla wiiCam.

Al fine di ridurre la quantità di dati inviata sul canale seriale, abbiamo raggruppato le informazioni sui sensori di prossimità e sull'acquisizione dell'arma in un solo byte, di cui ogni bit rappresenta lo stato di un sensore, secondo la *Figura 5.17*. Tale lettura viene effettuata con una frequenza di circa 50 Hz.

Inoltre, il colore restituito dal sensore è stato convertito in scala di grigi mediante il metodo della media, così che esso venga identificato mediante un

Figura 5.17: Struttura del byte rappresentante lo stato dei sensori

byte, anziché tre. Tale lettura è separata da quella degli altri sensori, in quanto viene effettuata con frequenza inferiore (circa 4 Hz), a causa dell'elevata necessità di risorse richieste.

Capitolo 6

L'applicazione del gioco

KartBot è corredato da un'applicazione realizzata in Java, che rappresenta il fulcro del suo funzionamento, attraverso il quale tutte le dinamiche del gioco prendono vita. Essa costituisce, come già spiegato nel *Capitolo 5,* il server di un sistema, di cui i kart sono i client.

L'applicazione non necessita di installazione e non sono richiesti requisiti particolari per il computer su cui viene utilizzata, ma è sufficiente che questo sia collegato alla rete WiFi, con il firewall opportunamente configurato per ricevere connessioni in ingresso.

Dal punto di vista grafico, l'applicazione presenta un'interfaccia semplice ed intuitiva, in modo da risultare di facile utilizzo per tutti.

6.1 Collegamento alla postazione di partenza

L'applicazione, per il corretto svolgimento del gioco, necessita di sfruttare una scheda Arduino appositamente montata sulla postazione di partenza, allo scopo di implementare le funzionalità di accensione del semaforo e di rilevamento del giro. Per la loro concreta realizzazione abbiamo instaurato un canale su porta seriale sempre attivo tra Arduino e l'applicazione stessa, al fine di permettere lo scambio di messaggi tra essi.

Le luci del semaforo sono costituite da led ad alta luminosità di colore rosso, ognuno dei quali è collegato ad un ingresso digitale di Arduino. Per replicare il funzionamento del semaforo presente nella gare automobilistiche, essi vengono accesi in sequenza, ad intervalli di 1,5 s. Trascorso un tempo di 2 s, in cui tutti i led sono accesi, essi si spengono e la gara ha inizio.

Per il rilevamento del completamento di un giro da parte del kart, abbiamo deciso di utilizzare due sensori di prossimità ad infrarossi, montati a lato della struttura metallica, in modo che siano contrapposti l'uno all'altro. Inizialmente avevamo pensato di far sì che fosse il kart stesso a riconoscere di aver effettuato un giro, soluzione sicuramente migliore perché permetteva di sapere esattamente di quale si trattasse, cosa non possibile invece con l'utilizzo dei sensori, ma troppo dispendiosa, in quanto la ricerca della scacchiera all'interno dell'immagine comportava un notevole impiego di risorse. Essendo solo due i kart presenti, risulta comunque possibile discriminare quale di essi ha completato il giro.

6.2 Struttura dell'applicazione

L'organizzazione delle classi all'interno dell'applicazione è pensata per evidenziare la suddivisione logica delle funzionalità che in essa sono implementate. Nell'*Appendice C* è riportato lo schema UML di come è stata strutturata.

Lo stato del gioco è mantenuto all'interno della classe *GameManager*, che è implementata come un singleton. In esso sono contenute tutte le strutture dati necessarie allo svolgimento della gara. Per la comunicazione con i kart abbiamo scelto di mantenere un elenco di sessioni socket attive in quel momento, univocamente definite, in modo da poter poi essere utilizzate dall'applicazione per lo scambio di messaggi con lo specifico client. La gestione di tali messaggi è demandata a tre listener, ognuno dei quali possiede un compito specifico. Per implementarli abbiamo scelto di utilizzare tre thread, così che potessero essere in esecuzione in contemporanea al flusso principale dell'applicazione, rimanendo costantemente in ascolto sull'apposita porta, in modo da ricevere i comandi di loro interesse ed elaborarli, chiamando le specifiche funzioni. I tre listener, presenti in un'unica istanza durante l'esecuzione dell'applicazione, sono, nel dettaglio:

- *CommandListener*: si occupa di accettare le connessioni basate su protocollo TCP/IP, mediante le quali l'applicazione scambia messaggi con i kart. Alla ricezione di questi, il listener decodifica i comandi che vi sono contenuti tramite l'utilizzo di espressioni regolari e, a seconda della tipologia, li gestisce nel modo opportuno;

Figura 6.1: Schermata del titolo

- *StreamingListener*: il suo compito è quello di ricevere lo streaming video proveniente dal kart telecomandato e di visualizzarlo nell'apposito spazio della finestra dell'applicazione. La comunicazione, in questo caso, è basata su protocollo UDP, in quanto è più rapido e il livello di affidabilità della trasmissione necessario non è elevato;

- *SerialListener*: è responsabile della comunicazione con Arduino, utilizzato per l'accensione del semaforo e la rilevazione del compimento di un giro da parte dei kart.

Per quanto riguarda, invece, l'aspetto grafico dell'applicazione, la classe responsabile è rappresentata dal singleton *GameGUI*. L'interfaccia è costituita da un'unica finestra con quattro diversi pannelli che vengono visualizzati o nascosti a seconda delle necessità, al fine di garantire la massima fluidità nel passaggio da una schermata di gioco all'altra.

6.3 Preparazione della gara

All'avvio dell'applicazione vengono subito create le classi Java essenziali per la gestione della logica del gioco. All'utente compare, invece, la schermata del titolo, dalla quale è possibile procedere alla fase di preparazione della gara vera e propria (*Figura* 6.1).

6.3.1 Connessione dei kart

Un'operazione essenziale è costituita dall'apertura delle sessioni con i client, alle quali l'applicazione non pone un limite, in quanto è ideata pensando all'eventualità di un'espansione futura che renda possibile la presenza di più di due kart. Al momento della connessione di un client viene creata una nuova sessione, che viene mantenuta fino al momento della disconnessione. Essa lo rappresenterà in modo univoco e verrà utilizzata per tutti gli scambi di messaggi necessari allo svolgimento del gioco. L'oggetto *Sessione* si occupa di rimanere costantemente in ascolto sulla porta, al fine di ricevere i comandi e gestirli nel modo corretto, mediante la creazione di un oggetto *CommandHandler* all'arrivo di ciascuno di essi. Il comando ricevuto viene da esso analizzato, con l'utilizzo di espressioni regolari, per riconoscere la tipologia a cui appartiene: *Login*, *WeaponTaken*, *WeaponUsed* o *Overtaking*. Tali classi sono tutte estensione di una classe astratta *Command* e ognuna verrà gestita nel modo opportuno, chiamando le funzioni necessarie. Il primo messaggio che viene inviato dal kart è la richiesta di login, nel formato:

```
@online,id=([a-zA-Z0-9]+),auto=(true|false)
```

dove *id* è il nome con cui viene identificato il kart e *auto* assume valore vero se il kart è autonomo, falso se è comandato. Il nome verrà utilizzato per associarlo alla sessione prima aperta, rendendone più semplice la ricerca in momenti successivi. Alla ricezione di tale comando, il kart viene aggiunto alla lista degli utenti online, dopo aver controllato che non ne esista già uno con lo stesso identificativo. La lista dei giocatori online è di tipo *OnLineUsersContainer,* un'estensione della classe astratta *UsersContainer* contenente oggetti di tipo *User*, e viene mantenuta all'interno dello stato del gioco. In tale contenitore sono presenti funzioni che permettono l'aggiunta, la rimozione e la ricerca di determinati utenti, mediante il loro username. La lista viene visualizzata nella parte sinistra della schermata (*Figura* 6.2) e, anche in questo caso, non è presente un limite nel numero dei giocatori che possono essere online in un determinato istante.

6.3.2 Selezione dei concorrenti e avvio della gara

Per poter iniziare la gara occorre, innanzitutto, selezionare i concorrenti che vi parteciperanno tra quelli online e stabilire le posizioni di partenza.

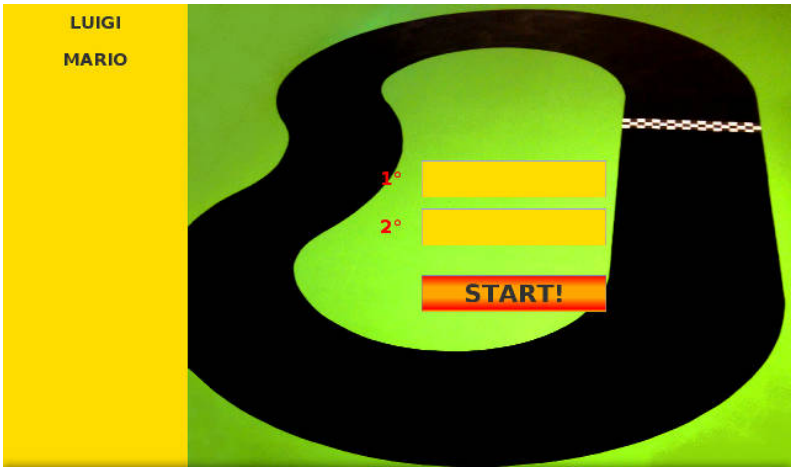


Figura 6.2: Connessione kart

Il procedimento è semplificato dal fatto che i nomi possono essere trascinati all'interno delle celle predefinite (*Figura 6.3*).

Il numero di giri attualmente previsto dall'applicazione è quattro e il numero massimo di giocatori è due, ma essa è predisposta in modo da poterli in futuro incrementare, ragionevolmente fino ad un massimo di quattro, date le dimensioni della pista e dei kart. Una volta avviata la gara mediante l'apposito pulsante, i giocatori scelti vengono inseriti nella lista dei *RacingUsers*, anche questa estensione della classe astratta *UsersContainer* e contenente quindi oggetti di tipo *User*, ma con funzioni in più, che permettono di operare su dati importanti per la gara, quali la posizione, il numero di giri e l'arma in possesso. L'inserimento avviene dopo aver effettuato alcuni controlli sull'esistenza dei giocatori selezionati tra quelli online e sull'assenza di duplicati e, se vi sono situazioni scorrette, la schermata viene ripresentata e l'inserimento va effettuato nuovamente. In caso contrario, l'applicazione invia ad Arduino, montato sulla postazione di partenza, il comando per l'accensione del semaforo, codificato con la stringa "s". Alla ricezione da esso del messaggio:

```
@countdown_expired
```

indicante il termine del conteggio alla rovescia, l'applicazione invia a tutti i giocatori il messaggio:

```
@start_race,nlaps=4,nplayers=([1-2]),position=([1-2])
```

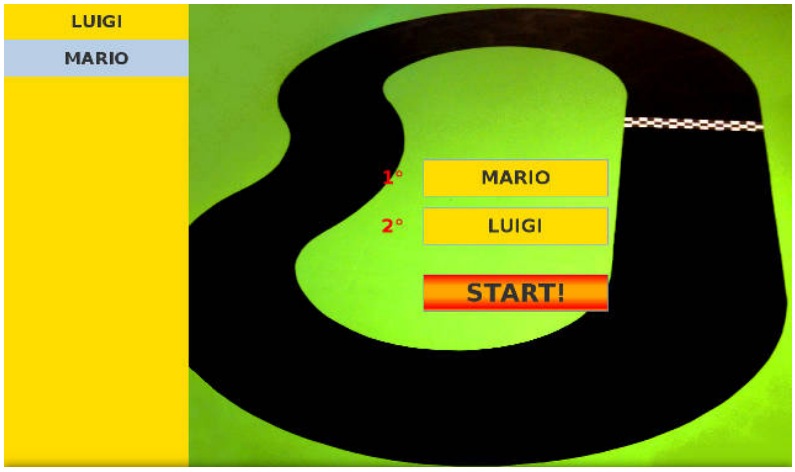


Figura 6.3: Selezione concorrenti

dove *nlaps* è il numero di giri previsto per la gara, *nplayers* è il numero di giocatori che stanno giocando e *position* è la posizione che è stata assegnata mediante l'inserimento negli spazi specifici. È prevista la possibilità di giocare anche da soli, nel caso lo si desideri.

6.4 Svolgimento della gara

Durante la gara sullo schermo vengono visualizzate tutte le informazioni importanti per il suo svolgimento. Abbiamo scelto di farlo, però, unicamente per i kart controllati, in quanto anche nel videogame avviene la stessa cosa. Quindi, nel caso vi sia un unico giocatore in gara contro l'intelligenza artificiale, verrà visualizzata un'unica schermata, mentre se si fronteggiano due giocatori la schermata sarà automaticamente suddivisa in due parti. L'applicazione è già predisposta per visualizzare fino ad un massimo di 4 finestre, nel caso in cui il numero dei kart venga incrementato. È essenziale, comunque, che lo schermo sia posto in una posizione ben visibile durante lo svolgimento della gara. Le informazioni mostrate su di esso, per ciascun giocatore, comprendono: la visualizzazione dello streaming video di quello che viene visto dal kart durante la gara, l'aggiornamento della posizione attuale in base ai sorpassi e del giro corrente e la visualizzazione delle armi in possesso (*Figura* 6.4).

Per semplicità, gli aspetti della grafica associati ai giocatori sono stati inclusi in una struttura dati di tipo *UsersRaceInfoContainer*, contenente un og-

Figura 6.4: Schermata gara

getto *UserRaceInfo* per ognuno di essi. Inoltre l'applicazione si occupa anche di realizzare gli effetti delle armi svantaggiose sul kart avversario.

6.4.1 Gestione dello streaming

La gestione dello streaming per i kart comandati è demandata allo *StreamingListener*. Esso è un thread che si occupa di rimanere in ascolto dei pacchetti di tipo UDP in arrivo. Alla ricezione di uno di questi, dopo aver riconosciuto il kart a cui appartiene, lo converte in un'immagine mediante la chiamata di apposite funzioni. Il risultato di tale procedimento viene poi visualizzato all'interno dello spazio dedicato nella finestra.

6.4.2 Gestione dei sorpassi

La segnalazione di un sorpasso è gestita passivamente, cioè è il kart sorpassato a comunicare all'applicazione il verificarsi dell'evento, mediante l'invio del comando:

```
@overtaken,id=([a-zA-Z0-9]+),invisible_opponent=(true|false)
```

dove l'*id* è l'identificativo del kart che è stato sorpassato e *invisible_ opponent* è una variabile pensata in previsione di un'espansione futura, in modo da indicare anche la presenza di un avversario sul lato opposto, per considerare anche i

sorpassi multipli. Per quanto riguarda la situazione attuale, questa variabile non è utilizzata. All'arrivo di un messaggio di questo tipo il *CommandHandler* riconosce un comando di tipo *Overtaking* e lo gestisce nel modo corretto. La posizione del kart che ha inviato la stringa viene aggiornata, scambiandola con quella dell'avversario che fino ad un attimo prima era immediatamente dietro di lui. L'aggiornamento sarà visualizzato anche all'interno della grafica dell'applicazione. A seguito di esso ad ognuno dei concorrenti viene inviato un messaggio con il formato:

```
@update_position,pos=([1-2])
```

dove *pos* indica la posizione attuale del kart destinatario. In questo modo la classifica viene mantenuta costantemente aggiornata e i kart sono a conoscenza della loro posizione corretta in ogni momento.

6.4.3 Gestione delle armi

La selezione della tipologia di arma viene effettuata a bordo del kart, al fine di evitare l'invio di due messaggi, uno per la richiesta e uno con la risposta. Il comando ricevuto dall'applicazione contiene già, quindi, la scelta effettuata ed è così definito:

```
@weapon_taken,id=([a-zA-Z0-9]+),weapon=type

con type=(IMMUNITY|LIKE_A_SHOT|VIBRATION|STOP|THUNDER)
```

dove *id* è l'identificativo del kart che ha ottenuto l'arma e *weapon* contiene la tipologia di arma che esso ha selezionato. Alla ricezione del messaggio l'applicazione memorizza l'arma a disposizione associata al kart e la visualizza in primo piano nella schermata, in modo che sia ben visibile al giocatore (*Figura* 6.5).

Quando quest'ultimo decide di utilizzarla, premendo l'apposito pulsante sul Wii Remote, il kart invia all'applicazione un messaggio del tipo:

```
@weapon_used,id=([a-zA-Z0-9]+),weapon=type

con type=(IMMUNITY|LIKE_A_SHOT|VIBRATION|STOP|THUNDER)
```

Figura 6.5: Possesso arma

dove *id* è l'identificativo del kart che l'ha inviato e *weapon* è la tipologia di arma. Dopo che è stato controllato l'effettivo possesso dell'arma, si possono verificare due situazioni:

- nel caso di armi vantaggiose, cioè *Immunità* e *Razzo*, il messaggio ricevuto costituisce una semplice notifica dell'avvenuto utilizzo;
- nel caso di armi svantaggiose, cioè *Vibrazione*, *Stop* e *Fulmine*, l'applicazione invia al kart avversario il comando che ne indica l'utilizzo su di esso:

`@exploiting_weapon,(VIBRATION|STOP|THUNDER).`

L'immagine dell'arma a questo punto scompare dalla finestra e ritorna indefinito il valore della variabile che ne indica il possesso da parte del kart.

6.4.4 Gestione dei giri

I valori letti dai sensori montati sulla postazione di partenza vengono analizzati mediante l'utilizzo di Arduino. Esso identifica il passaggio di un kart quando la distanza a cui almeno uno dei sensori percepisce la presenza di un ostacolo è inferiore alla metà della larghezza del tracciato. Se entrambi i sensori ricadono in questa condizione, viene distinto il caso in cui avviene il passaggio di un unico kart nel mezzo del tracciato da quello contemporaneo di due kart, controllando la coerenza della somma delle misure rilevate con la larghezza

Figura 6.6: Classifica finale

della pista. In particolare, se la differenza tra la larghezza di questa in centimetri e la somma dei valori misurati risulta uguale alla larghezza in centimetri di un kart con una tolleranza di ± 3 cm, viene identificato un unico passaggio, viceversa due. Data la non eccellente qualità dei sensori, abbiamo ritenuto opportuno identificare l'evento "passaggio di un kart" quando la rilevazione della presenza dell'ostacolo rimane costante per tre letture consecutive, effettuate a distanza di 80 ms l'una dall'altra.

Al riconoscimento dell'avvenuto passaggio, Arduino invia all'applicazione, attraverso la porta seriale, il messaggio:

```
@lap_completed
```

Nel caso di transito contemporaneo di due, le stringhe inviate sono due. Il *SerialListener*, costantemente in attesa sulla porta, all'arrivo del messaggio crea un oggetto *CommandHandler*, che riconosce un comando di tipo *LapCompleted*, anche questo implementazione della classe astratta *Command*, e lo gestisce, incrementando il conteggio del numero di giri del kart corretto. Tale informazione viene aggiornata anche all'interno della schermata della grafica.

6.5 Conclusione della gara

All'aggiornamento del numero di giri effettuati dal kart, se uno di essi ha raggiunto il totale previsto di quattro, la gara termina e sulla schermata compare la classifica finale (*Figura 6.6*).

L'interruzione del gioco può comunque avvenire manualmente in qualsiasi momento, in caso di problemi, mediante la pressione dell'apposito pulsante. In entrambi i casi, ai kart viene inviato un messaggio indicante il termine della gara:

```
@end_race
```

Dalla finestra conclusiva è possibile avviare una nuova gara, utilizzando il bottone presente su di essa.

Capitolo 7

Test e risultati

Al termine della creazione di KartBot, abbiamo effettuato alcuni test, di seguito elencati, al fine di fornire una valutazione dei risultati ottenuti. Per ognuno di essi sono riportate le funzionalità che ha interessato, la descrizione della prova fatta, situazioni problematiche rilevate durante il suo svolgimento e la sua effettiva riuscita. I primi test sono stati realizzati per sperimentare le funzionalità in modo separato, mentre l'ultimo ha come oggetto l'intero gioco. Per concludere, abbiamo riportato una valutazione complessiva del gioco in tutti i suoi aspetti.

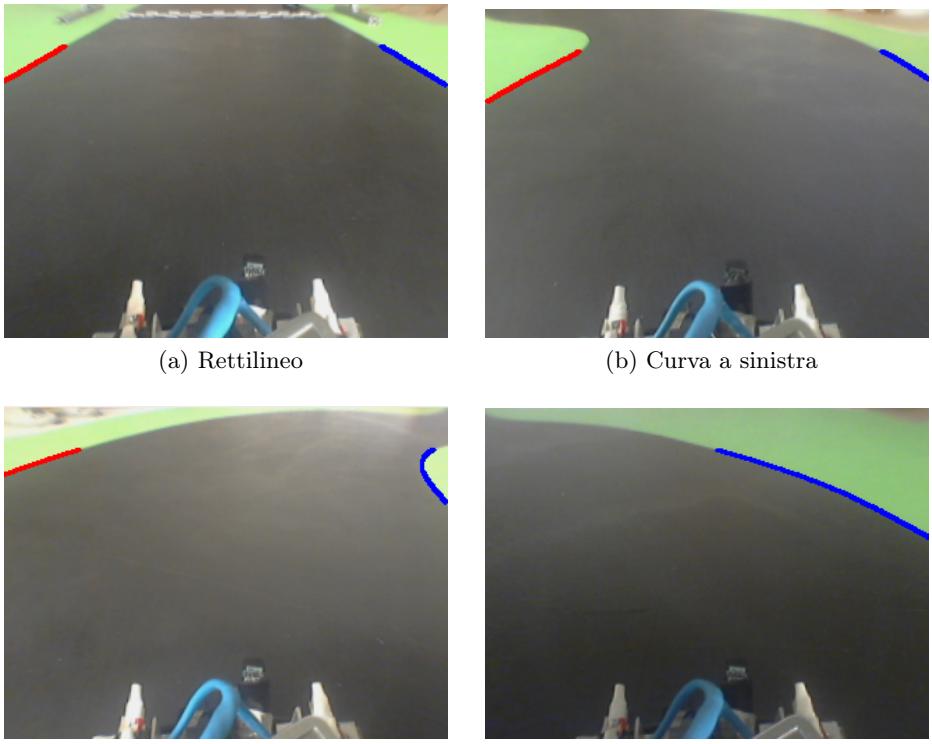
7.1 Test e risultati

7.1.1 Riconoscimento dei bordi

Per testare la capacità del kart autonomo di riconoscere la tipologia dei bordi del tracciato su cui sta viaggiando, lo abbiamo posizionato in più punti della pista, mantenendolo fermo. Come si può osservare in *Figura 7.1*, il bordo destro viene evidenziato con il colore blu, mentre quello sinistro con il rosso. In tutte e quattro le situazioni la tipologia viene riconosciuta correttamente, anche nel caso in cui nell'immagine sia presente solo uno dei due bordi (*7.1d*).

Da questa prova è emerso che il kart autonomo è in grado di identificare correttamente i limiti della pista e la loro forma, il che gli permettere di distinguere la situazione in cui si trova e di agire, quindi, di conseguenza.

Un problema riscontrato è il rilevamento, seppur in rari casi, di linee esterne al tracciato. Ciò si verifica nelle situazioni in cui fuori dalla pista sono presenti



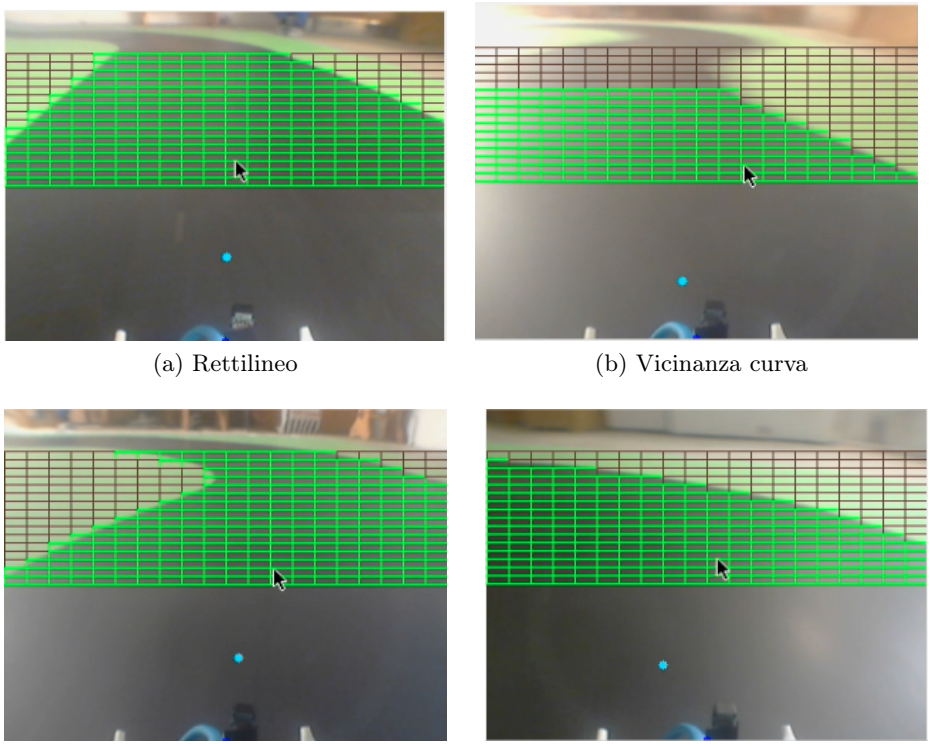
(a) Rettilineo

(b) Curva a sinistra

(c) Curva a destra

(d) Bordo unico

Figura 7.1: Riconoscimento bordi



(a) Rettilineo (b) Vicinanza curva

(c) Inizio curva (d) Curva

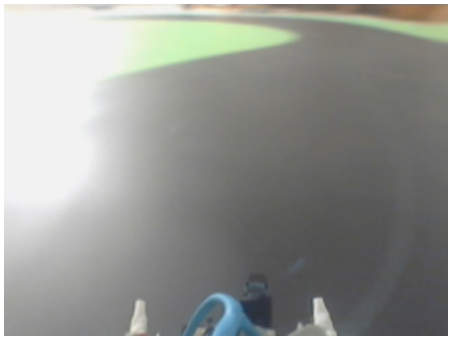
Figura 7.2: Riconoscimento tracciato

colori simili a quelli al suo interno. Per limitare i riconoscimenti errati viene fatta una verifica di coerenza sui bordi trovati, mantenendo unicamente il bordo destro più a destra, il sinistro più a sinistra e controllando anche la loro posizione rispettiva. Purtroppo in alcuni casi sfortunati in cui l'unico bordo rilevato è quello esterno alla pista si possono verificare comportamenti anomali del kart, solitamente corretti all'immagine successiva.

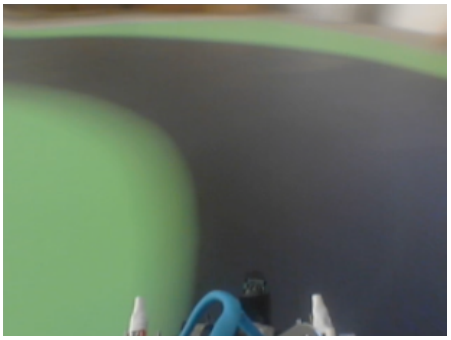
7.1.2 Gestione della navigazione

Per valutare l'abilità del kart autonomo di rimanere in pista, gli abbiamo fatto percorrere il tracciato, senza la presenza dell'avversario al suo interno. Come si può vedere in *Figura 7.2,* la forma della pista viene riconosciuta esattamente e la direzione è calcolata in modo corretto, come indicato dal pallino azzurro che rappresenta il goal dopo l'applicazione della rettifica.

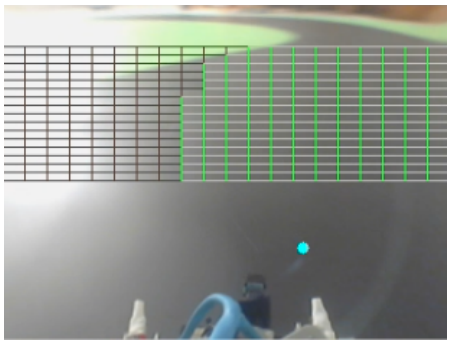
Come facilmente riscontrabile, osservando in particolare la *Figura 7.2b*,



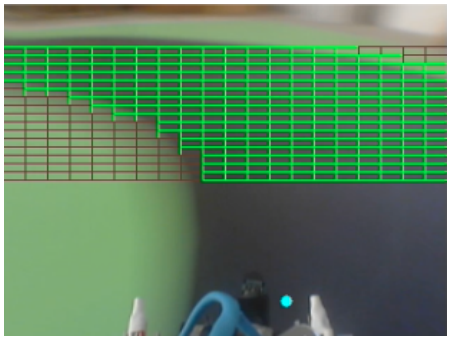
(a) Riflesso luminoso



(b) Sfocatura



(c) Approssimazione con riflesso luminoso

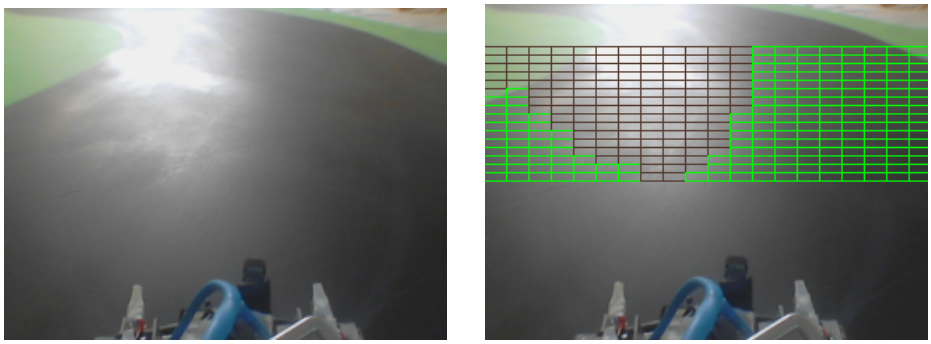


(d) Approssimazione con sfocatura

Figura 7.3: Situazioni sfavorevoli

durante i test sono emersi due problemi principali per la navigazione basata unicamente sul sistema di visione. Il primo è costituito dalla presenza di riflessi di luce particolarmente intensi e fastidiosi in alcuni punti della pista, che alterano i colori, disturbandone il riconoscimento, come si può vedere dalla *Figura 7.3a*. Il secondo è rappresentato dalla sfocatura delle immagini dovuta al movimento e alle vibrazioni della struttura, come appare chiaro dalla *Figura 7.3b*. Anch'essa rende estremamente difficoltosa la rilevazione dei bordi, a causa della mancanza di definizione delle immagini.

Entrambe le situazioni sfavorevoli esposte non sono del tutto eliminabili, ma abbiamo limitato i loro effetti negativi fornendo un livello più elevato di robustezza al sistema, ottenuto facendo sì che, nel caso di impossibilità di riconoscimento dei bordi con l'algoritmo principale, venga comunque riconosciuta una parte della pista. Ciò è possibile mediante la ricerca dei ROI che presentano una certa percentuale di pixel di colore nero, che vengono classificati come area libera su cui il kart può transitare. Come si può vedere in *Figura 7.3c*



(a) Riflesso centrale (b) Approssimazione con nero

Figura 7.4: Segmentazione della pista

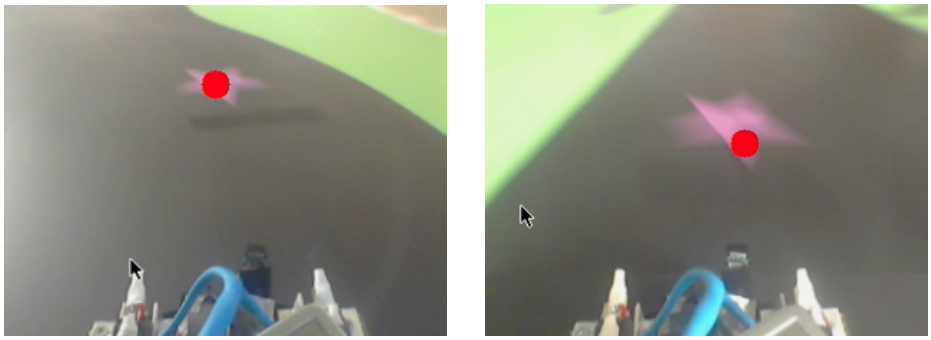
e *Figura 7.3d*, il risultato ottenuto nei due casi precedenti è un'approssimazione del tracciato che, anche se non presenta un livello elevato di precisione, consente comunque al kart di effettuare la scelta del movimento successivo da effettuare in modo non totalmente arbitrario.

Osservando la *Figura 7.4,* si può notare come in alcuni casi l'area venga segmentata in due parti a causa dell'intensa luce che fa apparire parti del tracciato simili al colore bianco. Ciò non risulta completamente eliminabile, ma non costituisce un grosso problema in quanto, anche se la pista non è riconosciuta interamente, si ha comunque un'informazione su dove essa è situata.

Vi sono poi situazioni sfortunate in cui la forma del tracciato, nonostante tutte le accortezze, non viene rilevata nel modo corretto. Questo si verifica principalmente in due casi: quando non viene riconosciuto nessun bordo e viene utilizzata l'approssimazione con il nero, ma l'immagine HSV utilizzata è di qualità molto scarsa, oppure quando viene riconosciuto solo uno dei due bordi e quindi le aree libere vengono posizionate partendo da esso e arrivando all'inizio, o alla fine a seconda dei casi, dell'immagine, facendo apparire la pista più larga di quanto non sia in realtà.

7.1.3 Acquisizione dell'arma

La capacità di acquisire l'arma è stata verificata posizionando il kart autonomo sulla pista, su cui erano presenti le quattro armi, senza la presenza dell'avversario.



(a) Arma lontana (b) Arma vicina

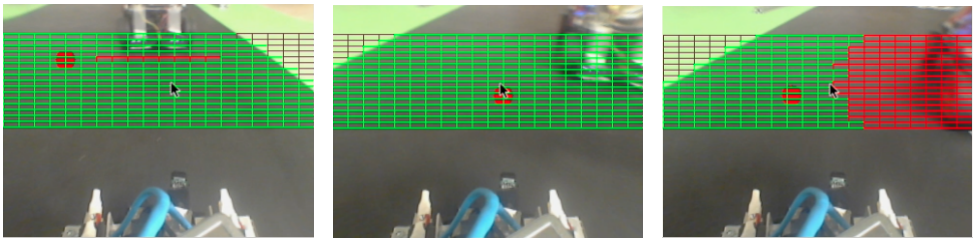
Figura 7.5: Goal con arma

Come si può notare dalle immagini, il kart è in grado di riconoscere le armi e di posizionare il goal esattamente al centro di quella più vicina. Successivamente esso riesce a raggiungerla mediamente nell'80% dei casi e, allo scattare degli interruttori appositi, notifica correttamente all'applicazione il suo conseguimento.

Anche in queste immagini è chiaro come il movimento sia un problema notevole nel riconoscimento dell'arma, in quanto può causare deformazioni evidenti nella forma della stella, impedendone la corretta identificazione.

7.1.4 Azione dell'arma

Per verificare la corretta azione delle armi sull'avversario abbiamo effettuato tre diversi test, uno per ciascuna tipologia di arma svantaggiosa, utilizzando entrambi i kart: quello controllato in pista e quello autonomo fermo all'esterno. Ogni prova è stata effettuata azionando manualmente gli interruttori per simulare l'acquisizione. Alla ricezione della notifica da parte dell'applicazione, è stata correttamente visualizzata l'icona riportante la tipologia di arma. Successivamente, all'utilizzo automatico di essa da parte del kart, l'applicazione ha ricevuto il messaggio atteso e inviato al kart controllato il comando che indica l'azione dell'arma su di esso. L'effetto è stato in tutti i casi applicato correttamente, sia dal punto di vista della tipologia sia della durata.



(a) Rilevamento dell'avversario (b) Avvicinamento (c) Avversario affiancato

Figura 7.6: Sorpasso - Primo esempio

7.1.5 Rilevazione del sorpasso

Il sorpasso è stato provato posizionando il kart autonomo fermo sulla pista. Al passaggio del kart controllato a lato, esso invia correttamente all'applicazione la notifica che indica l'avvenuto evento e quest'ultima aggiorna in modo esatto la posizione visualizzata.

Il problema di tale funzionalità è dato dalla qualità non eccellente dei sensori di prossimità utilizzati, che in alcuni istanti rilevano valori non coerenti con la situazione. Tale problema è stato parzialmente risolto suddividendo, come già spiegato nel *Capitolo 5*, il sorpasso in due fasi, il che lo rende più robusto rispetto a letture inesatte dei sensori. Un altro problema è dato dalla presenza di oggetti o persone all'esterno della pista che, se troppo vicine al kart, vengono rilevate come avversario. Questo è facilmente eliminabile, imponendo una zona di rispetto attorno alla pista, larga almeno quanto il tracciato stesso.

7.1.6 Gestione del sorpasso

Per verificare la capacità del kart autonomo di sorpassare quello controllato, abbiamo posizionato quest'ultimo fermo in pista. Un esempio di sorpasso è visibile in *Figura 7.6*: l'avversario viene rilevato e il goal è posizionato correttamente nell'area libera più ampia a disposizione (*7.6a*), successivamente ha inizio il sorpasso (*7.6b*), infine il kart è affiancato (*7.6c*), come denotato dall'indicazione dell'occupazione della pista dal lato in cui viene percepito.

Un problema che appare visibile nella sequenza mostrata è l'esistenza di un punto cieco che, nella fase intermedia del sorpasso, impedisce il riconoscimento dell'avversario. Anche se tale intervallo è stato ristretto dall'installazione dei

sensori di prossimità frontali, risulta ancora parzialmente presente. Questo è vero solo per un istante in quanto, subito dopo, i sensori posti frontalmente iniziano a percepire la sua vicinanza, evitando quindi la collisione. Altro problema è dovuto alla difficoltà di riconoscere l'avversario quando esso si trova molto spostato rispetto alla linea di vista della WiiCam, che in tali casi non riesce a rilevarlo. Durante i test sono emersi, inoltre, i problemi già riportati nel *Capitolo 5,* causati da situazioni di incertezza dovute al ristretto campo visivo, il quale può impedire la conoscenza della posizione all'interno della pista. Tali situazioni, seppur in parte risolte, in condizioni particolarmente sfavorevoli possono ancora essere problematiche e provocare un comportamento anomalo del kart.

7.1.7 Rilevamento del giro

La verifica del rilevamento del giro è stata effettuata facendo passare prima un solo kart sotto la struttura metallica e poi entrambi in contemporanea. In tutte e due le situazioni viene rilevato il numero effettivo di kart e viene inviato il corretto numero di messaggi mediamente nel 95% dei casi. Inoltre, il conteggio del giro viene aggiornato correttamente dall'applicazione.

Anche in questo caso, la non eccellenza della qualità dei sensori può provocare la rilevazione di valori non corretti, che è stata comunque limitata al minimo basando l'identificazione di un evento sulla lettura di più valori consecutivi. Inoltre, abbiamo notato che i sensori non possono essere tra loro allineati, perché ciò provocherebbe interferenze fastidiose e conseguenti letture errate, ma devono essere posizionati ad altezze diverse e inclinati lateralmente con angolazioni differenti.

7.1.8 Il gioco

Nel test finale, il gioco nel suo complesso è stato sottoposto a verifica. Il primo problema subito emerso è stato l'iniziale utilizzo di una velocità troppo elevata che rendeva il compito del giocatore piuttosto complicato. Essa è stata quindi ridotta nei test successivi, limitando purtroppo la gamma di velocità disponibili, già inizialmente piuttosto ristretta per la stessa motivazione, rendendo meno visibili alcuni effetti delle armi e anche il rallentamento all'uscita della pista. È emersa, inoltre, la difficoltà nel rendere la velocità dei due

kart equivalente, a causa anche dei suoi valori molto variabili in condizioni di diverso livello di carica della batteria.

Come ovvio, anche dal test totale sono emerse le problematiche già singolarmente affrontate in quelli precedenti. Un nuovo problema riscontrato è invece dovuto alla possibilità per il kart controllato di "spingere" quello autonomo al di fuori della pista. Infatti, se durante la gara il giocatore posiziona il suo kart di fianco a quello avversario e lo sposta sempre più verso quest'ultimo continuando a mantenersi al suo fianco, restringe costantemente l'ampiezza dell'area libera a disposizione del kart autonomo e lo porta ad andare fuori dal tracciato. Tale situazione non è purtroppo risolvibile in alcun modo.

7.2 Valutazione finale

Nonostante le diverse problematiche emerse durante i test, alcune delle quali forse ulteriormente migliorabili mentre altre non risolvibili, il gioco ottenuto nel suo complesso è comunque piuttosto soddisfacente, in quanto molte di tali situazioni si verificano raramente o comunque non incidono sulla sua qualità.

KartBot è risultato essere, come desideravamo, un gioco molto intuitivo e di facile comprensione per tutti, seppur occorra prendere un minimo di dimestichezza con l'uso del controller, soprattutto quando il kart procede nel verso opposto a quello in cui si sta guardando, esperienza ben nota anche nei videogame, ma tanto più rilevante a causa di oggetti reali da controllare. La presenza delle diverse armi costituisce effettivamente un valore aggiunto, in grado di rendere la gara molto più avvincente e divertente. Anche l'impegno posto nella realizzazione di un ambiente realistico e di elevata qualità, unito alla creazione di un'applicazione semplice e graficamente piacevole e all'uso di luci e suoni per migliorare l'interattività costituiscono senza dubbio aspetti vincenti.

In conclusione, KartBot è un gioco appassionante e dinamico, in grado di coinvolgere il giocatore in gare sempre nuove e sorprendenti.

Capitolo 8

Conclusioni

I robogame rappresentano uno dei settori della robotica con le maggiori potenzialità di una vasta diffusione in ambiente domestico. Il suo studio è finalizzato alla realizzazione di giochi tali da poter attrarre l'interesse dei consumatori finali, utilizzando tecnologie avanzate e mantenendo, al contempo, i costi contenuti.

Scopo della tesi è stato quello di realizzare KartBot, un robogame consistente in una gara tra kart coinvolgente e interattiva, che ha trovato l'ispirazione per la sua ideazione nel famoso videogame Mario Kart Wii e nelle numerose robot racing esistenti. L'obiettivo che ci siamo posti è stato quello di realizzare un gioco che unisse la tradizionalità delle gare con le macchinine ad elementi altamente tecnologici appartenenti a discipline di nuova generazione, ottenendo un prodotto che incarna in sé la fusione tra passato e futuro.

L'aspetto innovativo è rappresentato dalla realizzazione di un kart totalmente autonomo in grado di gareggiare contro un giocatore umano ad una velocità elevata, sfruttando unicamente informazioni provenienti dal sistema visivo e dai sensori, senza conoscenza pregressa dell'ambiente. L'elaborazione avviene totalmente a bordo in tempo reale, per far fronte alla variabilità data dal contesto dinamico caratteristico della gara.

Notevole attenzione nello sviluppo di KartBot è stata rivolta a fornire la massima usabilità possibile a giocatori di qualunque età e a migliorare l'esperienza di gioco, mediante l'utilizzo di feedback per incrementarne l'interattività, come l'utilizzo di luci e suoni, e la creazione di un ambiente il più possibile realistico ed esteticamente di elevata qualità.

Dai test effettuati è emerso come il risultato ottenuto sia un gioco emozio-

nante e in grado di rapire l'attenzione, pur mantenendo un livello di complessità minimo, sia per quanto riguarda il regolamento sia le abilità necessarie per utilizzarlo. Senza dubbio esso può essere ancora migliorato, ad esempio rendendo la traiettoria del kart autonomo il più possibile simile a quella del giocatore umano, in modo da affrontare alcune situazioni in modo più intelligente. Altre problematiche che affliggono i sistemi di visione, come l'elevata dipendenza dai cambiamenti della luce, la sfocatura dovuta al movimento e il ristretto campo visivo, non sono invece del tutto risolvibili. Miglioramenti possono comunque essere ottenuti sostituendo la camera con una che abbia una qualità migliore, aggiungendone un'altra per aumentare l'ampiezza della porzione di tracciato inquadrata e illuminando la pista in modo uniforme, così che non risulti soggetta a variazioni problematiche della luminosità.

Tutte le decisioni prese nella realizzazione di KartBot sono state pensate per fornire la massima flessibilità e permettere una possibile espansione futura. Esso potrebbe, ad esempio, essere esteso ad un numero maggiore di kart, fino ad un massimo di quattro. Va notato che buona parte delle funzionalità sono già state implementate in modo da non essere vincolate alla presenza di due soli concorrenti. Inoltre, fornire a tutti i kart Odroid come unità di elaborazione permetterebbe di utilizzare ciascuno di essi sia come controllato sia come autonomo e gareggiare così secondo le diverse combinazioni possibili. Anche le tipologie di armi presenti potrebbero essere aumentate, rendendo il gioco più avvincente. Per quanto riguarda il tracciato, la creazione di altre piste di forme differenti, eviterebbe la perdita d'interesse, aggiungendo variabilità. Un'idea potrebbe essere anche quella di rendere la pista un grosso puzzle, così da poterla comporre ogni volta in modo diverso. Affinché esso possa effettivamente conquistare un posto in ogni casa, sarebbe necessario, però, pensare ad una miniaturizzazione di tutti i componenti. Altro aspetto da valutare è la possibilità di inserire più livelli di difficoltà per adeguare maggiormente il gioco alle caratteristiche dell'utilizzatore.

In conclusione, il risultato ottenuto, seppur sicuramente migliorabile, presenta già un livello di qualità tale da renderlo appetibile agli occhi di futuri giocatori. KartBot è un robogame appassionante, divertente ed intuitivo, punto d'incontro tra tradizione e innovazione. Proprio la sua semplicità, unita all'aspetto gradevole, lo rende adatto a tutte le fasce d'età e in grado di unire le generazioni: appare piuttosto facile immaginare nonni e nipoti fronteggiarsi in una sfida all'ultimo colpo su un campo di battaglia in livrea nera e verde.

Bibliografia

[1] A. Abruzzese. http://it.wikipedia.org/wiki/Alberto_Abruzzese/.

[2] Cultura del videogioco: studi e ricerche. http://www.aesvi.it/.

[3] Bill Gates. A robot in Every Home. *Scientific American Magazine*, 2007.

[4] AIRLab. http://airlab.elet.polimi.it/index.php/AIRWiki.

[5] D. Martinoia, D. Calandriello, and A. Bonarini. Physically Interactive Robogames: Definition and Design Guidelines. *Robotics and Autonomous System*, 61(8):739–748, 2013.

[6] Jedi Training. http://airwiki.ws.dei.polimi.it/index.php/Jedi_Training_Robogame.

[7] RoboTower. http://airwiki.ws.dei.polimi.it/index.php/RoboTower.

[8] Pac-Bot. http://airwiki.ws.dei.polimi.it/index.php/Pac-Bot.

[9] Mario Kart Wii. http://www.mariokart.com/wii/launch/.

[10] Intelligent Ground Vehicle Competition. http://www.igvc.org/.

[11] Scorpion. http://www.igvc.org/design/2013/California%20State%20University%20Northridge.pdf.

[12] International Autonomous Robot Racing Challenge. http://robotracing.wordpress.com/.

[13] C. Wang, P. Mukherjee, A. Das, G. Salas Bolanos, and N. Cavan. Design and Implementation of the University of Waterloo IARRC Entry. http://robotracing.files.wordpress.com/2010/08/2010-team-4-megalodon-uwar-university-of-waterloo-autonomous-racing.pdf.

[14] Eui-Jung Jung and Byung-Ju Yi. Task-oriented navigation algorithms for an outdoor environment with colored borders and obstacles. *Intelligent Service Robotics*, 6(2):69–77, 2013.

[15] Anki. `http://anki.com/`.

[16] Wii Remote. `http://www.nintendo.it/Wii/Accessori/Accessori-Wii-Nintendo-Italia-626430.html`.

[17] Raspberry Pi. `http://www.raspberrypi.org/`.

[18] Odroid U2. `http://www.hardkernel.com/main/products/prdt_info.php?g_code=G135341370451/`.

[19] Archlinux ARM. `http://archlinuxarm.org/`.

[20] Webcam C270. `http://www.logitech.com/it-it/product/hd-webcam-c270`.

[21] Sensore di prossimità Sharp 2d120x. `http://www.sharpsma.com/webfm_send/1205`.

[22] Arduino micro. `http://arduino.cc/en/Main/arduinoBoardMicro`.

[23] Sensore di colore ADJD-S311. `http://dlnmh9ip6v2uc.cloudfront.net/datasheets/Sensors/LightImaging/datasheetCR999.pdf`.

[24] Circuito WiiCam. `http://letsmakerobots.com/node/7752`.

[25] Iracer. `https://www.sparkfun.com/products/11162`.

[26] Libreria OpenCV. `http://opencv.org/`.

[27] Libreria LibSerial. `http://libserial.sourceforge.net/`.

[28] Y. M. Wang, Y. Li, and J. B. Zheng. A Camera Calibration Technique Based on OpenCV. *Information Sciences and Interaction Sciences (ICIS), 2010 3rd International Conference on*, pages 403–406, 2010.

[29] Calibrazione webcam usando la libreriaOpenCV. `http://docs.opencv.org/doc/tutorials/calib3d/camera_calibration/camera_calibration.html`.

[30] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2000.

[31] Algoritmo di Canny. `http://docs.opencv.org/doc/tutorials/imgproc/imgtrans/canny_detector/canny_detector.html`.

[32] Zhengyou Zhang. Single-View Geometry of A Rectangle With Application to Whiteboard Image Rectification. `http://research.microsoft.com/en-us/um/people/zhang/papers/whiteboardrectification.pdf`, 2013.

[33] Libreria bluetooth BlueZ. `http://www.bluez.org/`.

[34] Libreria Wii Remote Libcwiimote. `http://libwiimote.sourceforge.net/`.

Appendice A

Configurazione delle unità di elaborazione

In questa appendice analizziamo nel dettaglio le caratteristiche delle unità di elaborazione utilizzate, per poi descrivere la loro configurazione e quella dell'infrastruttura di rete.

A.1 Odroid

L'unità di elaborazione per il kart autonomo è il mini computer Odroid-U2, prodotto dall'azienda coreana HardKernel, di cui abbiamo riportato in *Figura A.1* lo schema a blocchi e in *Tabella A.1* le caratteristiche tecniche.

A.2 Raspberry Pi

Il kart controllato è equipaggiato con il mini computer Raspberry Pi, prodotto dall'inglese Raspberry Foundation, di cui abbiamo riportato le caratteristiche tecniche in *Tabella A.2.*

A.3 Configurazione

Entrambe le unità di elaborazione non prevedono né hard disk né unità a stato solido, ma si basano su una scheda SD per il boot e la memoria non volatile, adatta ad ospitare sistemi basati su kernel Linux. La configurazione di una nuova unità o la sostituzione della scheda SD può essere effettuata tramite

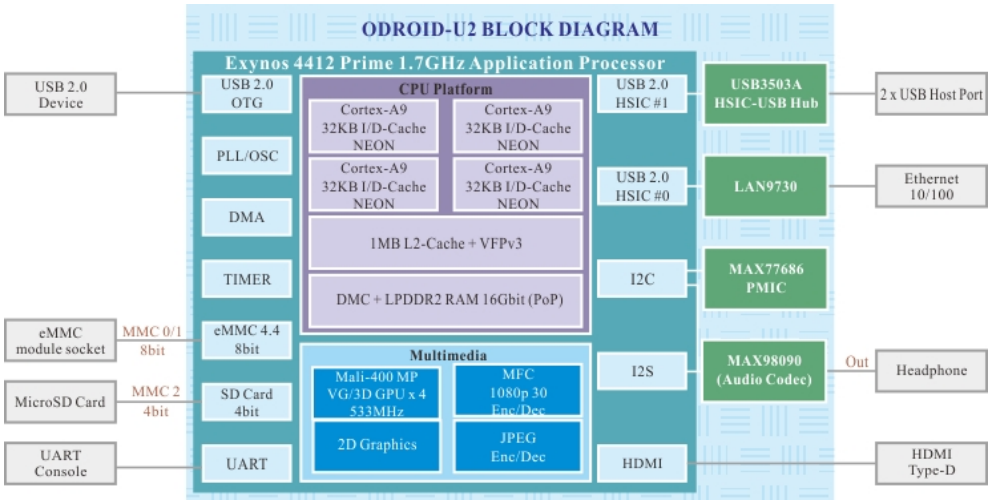


Figura A.1: Schema a blocchi Odroid

CPU	1.7GHz Exynos4412 Prime Cortex-A9 Quad-core processor with PoP (Package on Package) 2Gbyte LPDDR2 880Mega Data Rate
PMIC	MAX77686 Power Management IC from MAXIM
HSIC USB 2.0 hub	USB3503A Integrated USB 2.0-compatible hub / HSIC upstream port from SMSC
Ethernet	LAN9730HSIC USB 2.0 to 10/100 Ethernet controller with HP Auto-MDIX from SMSC
Audio Codec	MAX98090 is a full-featured and high performance audio CODEC from MAXIM
Protection IC	BQ24381Over-voltage, Over-current protection IC from TI
USB load switch	R5524N Protection IC for USB power supply from RICOH
HDMI conditioner	IP4791CZ12 HDMI transmitter interface protector with level shifter from NXP
HDMI connector	Standard Micro-HDMI, supports up to 1920 x 1080 resolution
I/O port	USB Host x 2, Device x 1, Ethernet RJ-45, Headphone Jack
Storage slot	Micro-SD slot, eMMC module connector
Microphone	DMO-B125T26-6P Digital MEMS Microphone (Omni-Directinal) from BSE
DC input	5V / 2A input, Plug specification is inner diameter 0.8mm and outer diameter 2.5mm

Tabella A.1: Caratteristiche tecniche Odroid

CPU	700 MHz ARM11 ARM1176JZF-S core
GPU	Broadcom VideoCore IV,OpenGL ES 2.0,OpenVG 1080p30 H.264 high-profile encode/decode
Video Out	Composite video | Composite RCA, HDMI (not at the same time)
Ethernet	10/100 wired Ethernet RJ45
Audio Out	TRS connector | 3.5 mm jack, HDMI
I/O low level	General Purpose Input/Output (GPIO) pins, Serial Peripheral Interface Bus (SPI), I^2C, I^2S[2], Universal asynchronous receiver/transmitter (UART)
I/O port	USB Host x 2, Device x 1, Ethernet RJ-45, Headphone Jack
Storage slot	Secure Digital|SD / MMC / SDIO card slot
DC input	5V / 2A input, Plug specification is inner diameter 0.8mm and outer diameter 2.5mm

Tabella A.2: Caratteristiche tecnciche Raspberry

una semplice copia dell'immagine *.img* da noi creata, scegliendo quella adatta alla piattaforma del caso, mediante il seguente comando:

```
$ dd /home/pathToimg/file.img /dev/pathToSD bs=1M
```

In essa sono contenuti tutti i file necessari al funzionamento ed è già pronta all'uso.

Tuttavia di seguito riportiamo le procedure utilizzate per la configurazione sia di Odroid sia di Raspberry, partendo dall'installazione del sistema operativo e arrivando alla configurazione di OpenCV, delle librerie per la comunicazione con il Wii Remote e per l'utilizzo della porta seriale.

La distribuzione ArchLinux ARM può essere trovata all'indirizzo web `http://archlinuxarm.org/`, da cui è possibile scaricare il file *.img* per l'unità di elaborazione desiderata, che va poi copiata sulla scheda SD usando il comando scritto sopra. In modo automatico, essa viene installata in una partizione da 2 GB, ridimensionabile tramite l'utilizzo di utility come *"GParted"* o simili.

L'installazione di OpenCV è stata effettuata attraverso il gestore dei pacchetti, avendo precedentemente installato *"pkg-config*", con il seguente comando:

```
$ pacman -S opencv pkg-config
```

Per la comunicazione bluetooth con il Wii Remote abbiamo usato la libreria *"bluez",* installabile attraverso il gestore dei pacchetti con il seguente comando:

```
$ pacman -S bluez
```

L'avvio del servizio bluetooth al boot del sistema viene attivato con i seguenti comandi:

```
$ systemctl  enable bluetooth.service
$ systemctl start bluetooth.service
```

La comunicazione con il Wii Remote è stata gestita attraverso la libreria *"libcwiimote"*, la quale richiede la presenza del solo pacchetto bluez. Non essendo presente nei repository ufficiali di ArchLinux, abbiamo dovuto procedere alla sua compilazione ed installazione, seguendo i comandi sotto elencati, dopo aver scaricato i file sorgente reperibili all'indirizzo `http://libwiimote.sourceforge.net/#download`. Dal momento che essa è scritta per essere utilizzata con il linguaggio C, abbiamo dovuto eseguire una *patch* (scaricabile dal sito `http://sourceforge.net/p/libwiimote/bugs/3/`) per permettere il link e la compilazione in C++.

```
$ tar -xzvf libwiimote-x.x.gz
$ cd libwiimote-x.x
 (copiare il file add_c++support_add_libcwiimote.pc_.
    patch
 nella cartella libwiimote-x.x)
$ patch -p1 <~/dev/libcwiimote/add_c++
   support_add_libcwiimote.pc_.patch'
    patch -p1 <add_c++support_add_libcwiimote.pc_.patch
$ autoconf -o configure configure.in
$ make
$ sudo make install
```

Durante il processo di compilazione ci siamo trovati ad affrontare un bug noto, che si manifesta con la presenza del seguente errore, dopo l'esecuzione del comando *"configure":*

```
configure: error: We require BlueZ
```

La soluzione consiste nel cambiare, in tutti i file sorgente, il nome della variabile *"hci_remote_name"* in *"hci_read_remote_name"*.

La comunicazione con Arduino su porta seriale è stata implementata usando le libreria *"libSerial"*, anch'essa non presente nella repository ufficiale di ArchLinux, e scaricabile da `http://libserial.sourceforge.net/`. La compilazione e l'installazione sono stata effettuate con i seguenti comandi:

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

Tutti i file sorgente sono contenuti in *"/home/mkb/mkb-project/trunk/src/client"* e possono essere compilati attraverso il comando *"make"*, eseguito da terminale in tale cartella. Il file eseguibile viene creato nella stessa posizione ed è presente un collegamento ad esso nella *"home"* dell'utente *"mkb"*.

Per quanto riguarda la connessione WiFi, essa è implementata usando un router wireless Cisco, creando una rete con SSID "*mkb-net*", protetta con cifratura WPA2 e chiave "*Mkb2013@polimi*". All'avvio del software di bordo, i kart si connettono automaticamente alla rete, tramite i comandi di seguito descritti, auto-assegnandosi un indirizzo IP statico della rete 192.168.1.0/24.

```
$ netctl enable mkb-net-wireless
$ netctl start mkb-net-wireless
```

dove *"mkb-net-wireless"* è il nome di un file, rappresentante un profilo di rete, salvato nella cartella *"/etc/netctl"*, contenente le seguenti righe:

```
Description='A basic static ethernet connection'
Interface=wlan0
Connection=Wireless
IP=static
Address=192.168.1.x/24
#Routes=('192.168.0.0/24 via 192.168.1.2')
Gateway=192.168.1.1
```

Entrambe le unità di elaborazione sono configurate in modo da avere sempre attivo un server ssh sulla porta 21, per un'eventuale connessione da terminale, ed un server VNC, sulla porta di default. Inoltre esse sono configurate in modo che la scheda di rete Ethernet, attivata solo al momento dell'accensione se presente un cavo di rete connesso, cerchi di acquisire un indirizzo IP, di una qualsiasi rete, tramite richiesta DHCP. L'utente utilizzato ha nome *"mkb"* e password *"Mkb2013"*, mentre il superutente *"root"* ha password *"root"*.

Appendice B

Istruzioni di montaggio

Il montaggio della struttura del kart, così come il posizionamento dei componenti, non richiede particolari considerazioni, ad eccezione della necessità di distribuire la maggior parte del peso nella parte posteriore.

Il discorso ovviamente è diverso nel caso delle interconnessioni tra i componenti hardware e Arduino: di seguito ne riportiamo gli schemi dettagliati per entrambi i kart. In *Figura B.1* è mostrato lo schema generale delle connessioni con Arduino. Per le periferiche che sfruttano la comunicazione I2C è stato necessario convertire i 5 V, erogati da Arduino, a 3,3 V. Per farlo abbiamo adottato due soluzioni differenti per i due kart: in quello autonomo abbiamo creato il circuito mostrato in *Figura B.2a,* mentre in quello controllato abbiamo realizzato una semplice rete resistiva, visibile in *Figura B.2b.* Tale diversità deriva dal fatto che, nel kart autonomo, la presenza della wiiCam creava problemi di assorbimento sulla rete resistiva. In *Figura B.3* abbiamo riportato lo schema circuitale per il collegamento della wiiCam.

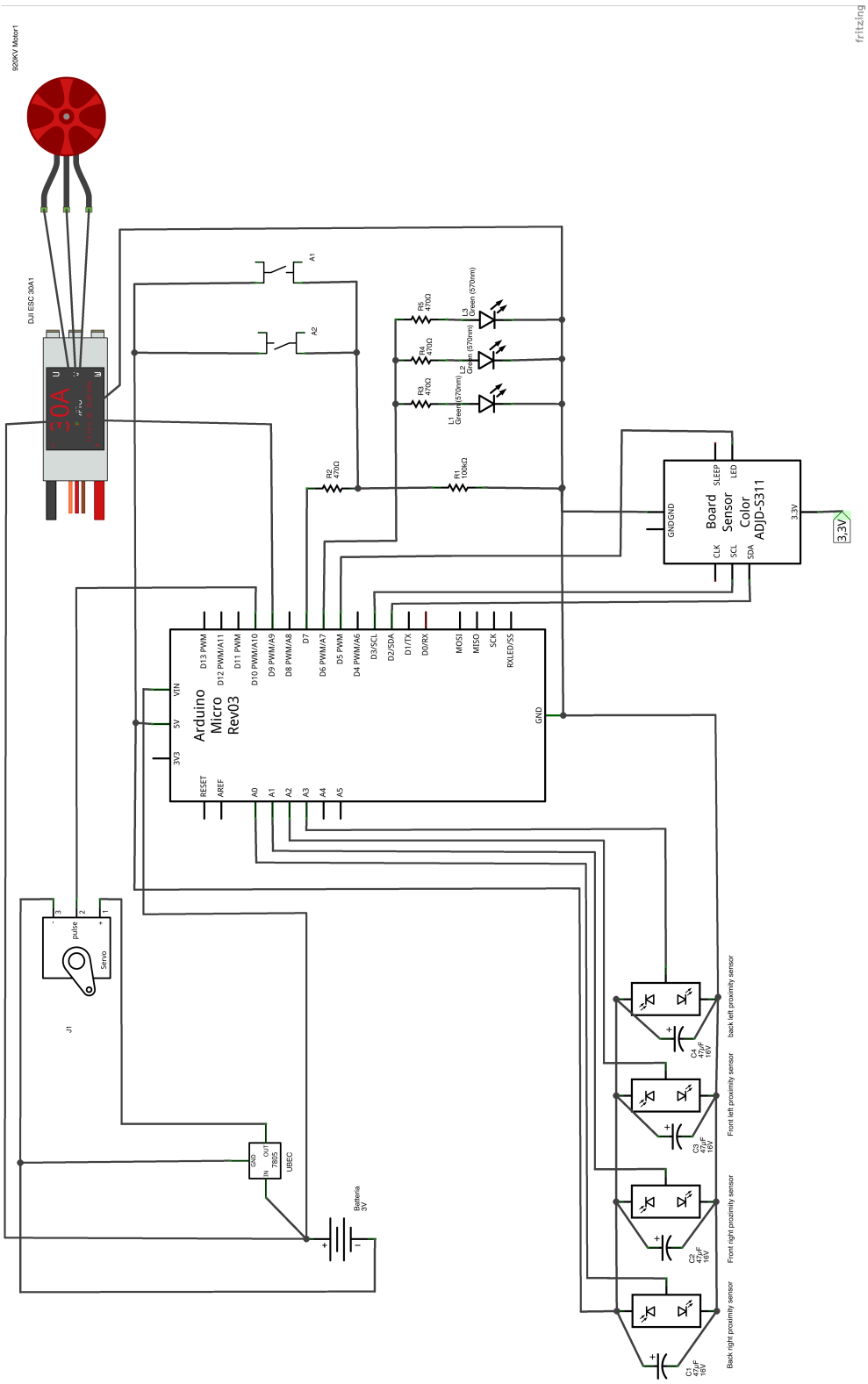
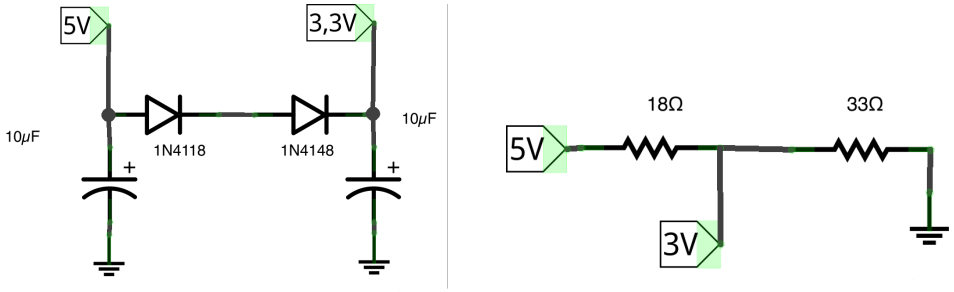


Figura B.1: Schema connessioni



(a) Kart autonomo (b) Alimentazione I2C kart controllato

Figura B.2: Alimentazione I2C

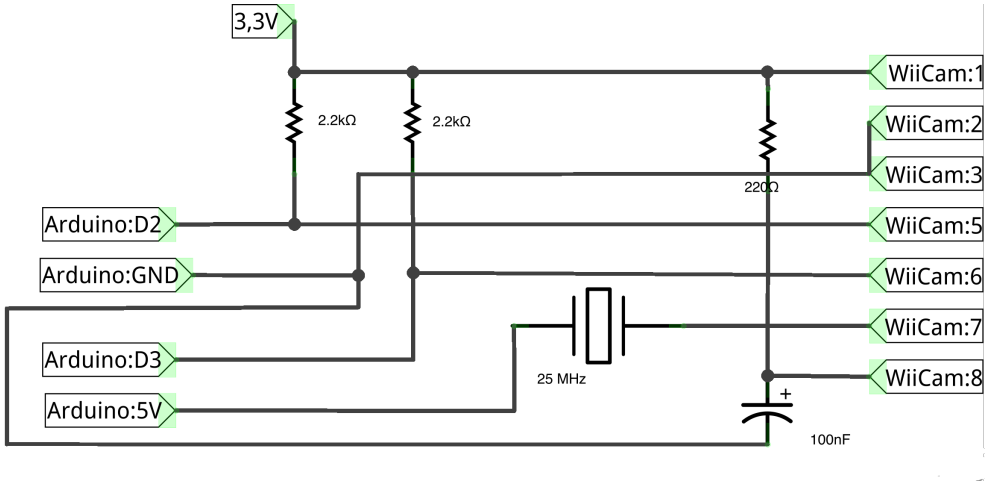


Figura B.3: Schema wiiCam

Appendice C

Documentazione del software

Di seguito riportiamo i diagrammi delle classi dell'applicazione eseguita sul computer remoto, divisi per package, e del software di bordo. In *Figura C.1* è mostrato il diagramma UML relativo ai package creati nell'applicazione centrale, in *Figura C.2* è presente il diagramma delle classi del package "command", contenente tutti i comandi gestiti dal software, in *Figura C.3* quello del package "data structure", contenente tutte le strutture dati utilizzate, e in *Figura C.4* quello del package "MKB_app", contenente la classe *Main*, i listener per le connessioni e gli oggetti necessari al loro utilizzo.

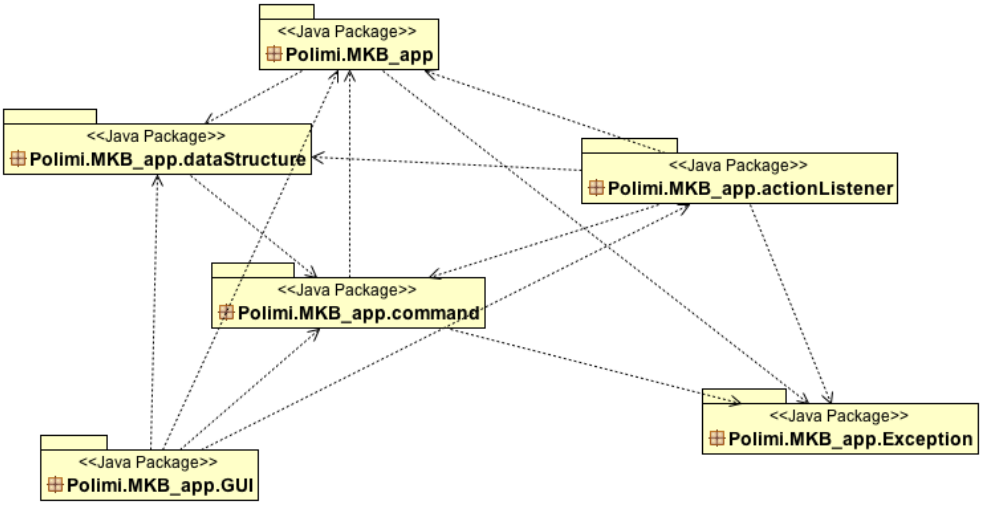


Figura C.1: Package UML

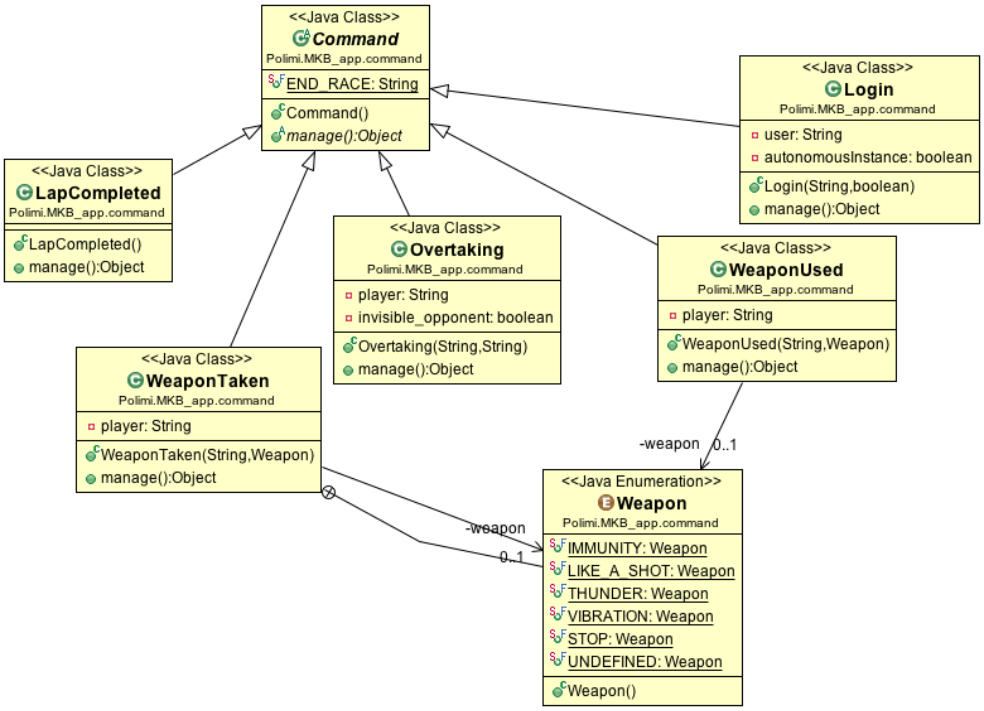


Figura C.2: Diagramma delle classi «command»

Figura C.3: Diagramma delle classi «data structure»

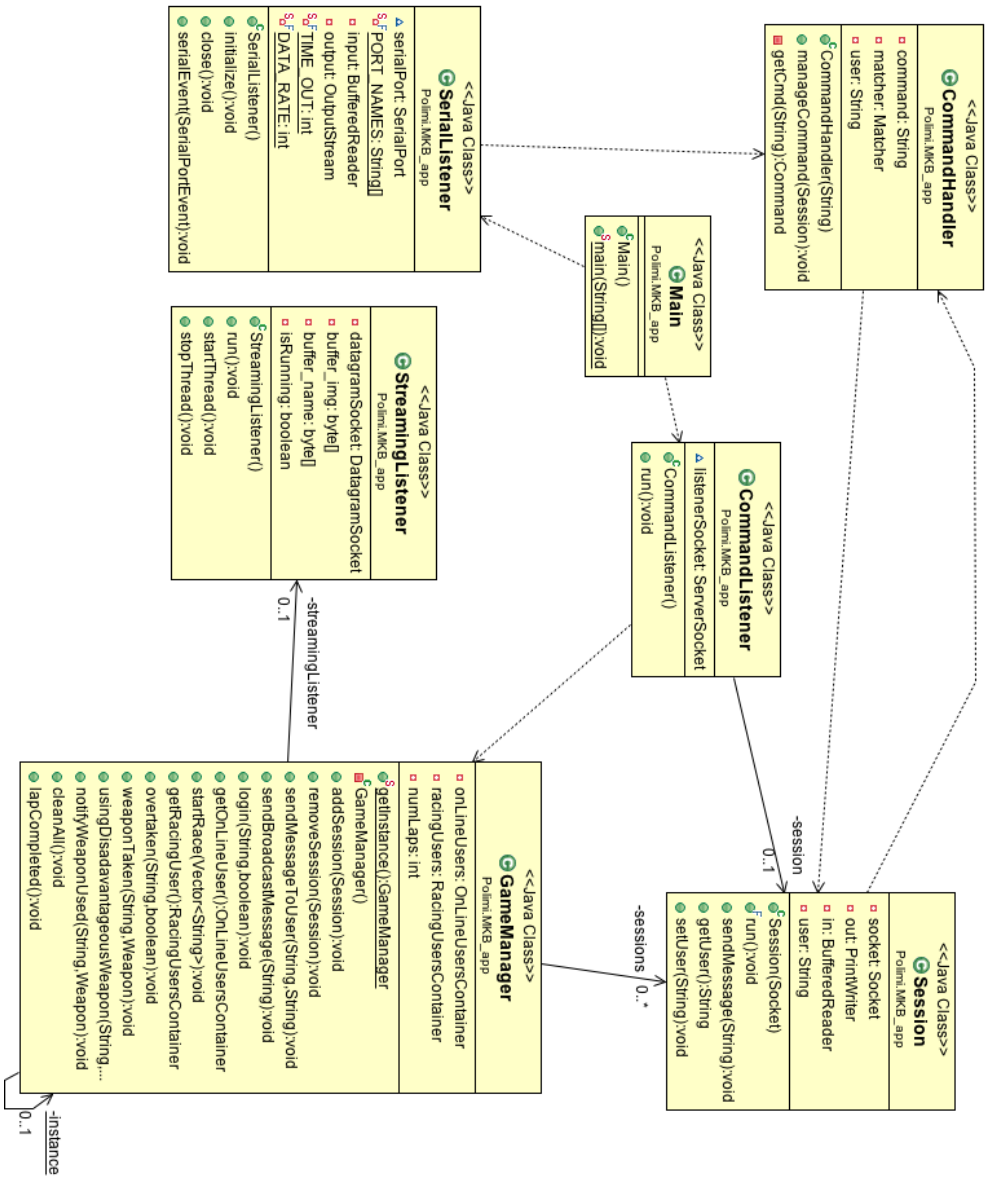


Figura C.4: Diagramma delle classi «MKB_app»

Appendice D

Manuale d'uso

KartBot permette di essere utilizzato in diverse configurazioni: due giocatori che gareggiano tra loro comandando due kart controllati oppure un giocatore umano che sfida l'intelligenza artificiale. È comunque sempre possibile avviare il gioco anche in presenza di un solo kart in pista, sia autonomo che non.

La preparazione necessaria per predisporre il gioco è semplice e attuabile da qualsiasi utente medio. Per iniziare il gioco occorre essere in possesso di tanti Wii Remote quanti sono i kart che si desidera controllare e dell'applicazione Java contenuta nel file "*KartBot.jar*". È indispensabile, inoltre, assicurarsi che il computer su cui essa sarà eseguita sia connesso alla rete WiFi *"Mkb-net*", configurata con le impostazioni descritte nell'*Appendice A*. All'avvio dell'applicazione principale viene mostrata la schermata del titolo, dalla quale, premendo il pulsante *"Play!",* si avvia una nuova sessione di gioco. Nella schermata successiva sono presenti uno spazio sulla sinistra, in cui compaiono i nomi dei kart attualmente connessi, e alcune caselle di testo, ognuna delle quali rappresenta una posizione di partenza. L'accensione del kart, effettuata tramite lo spostamento dell'apposito interruttore su ON (*Figura D.1*), non implica la sua connessione automatica all'applicazione. Per far sì che ciò accada è necessario collegarsi ad esso tramite una connessione ssh o vnc e avviare il software a bordo, eseguendo il file *"client",* presente nella cartella *"home"* dell'utente *"mkb*". Nel caso di kart controllato, durante tale procedimento è necessario tenere sempre premuti i tasti "1" e "2" del Wii Remote e fornire al programma "client", come parametro da linea di comando, l'indirizzo MAC del controller, visualizzabile tramite il comando *"hcitool scan*". Qualora l'unità di elaborazione non si connetta alla rete WiFi in automatico, è consigliabile un riavvio del

Figura D.1: Interruttore ON/OFF

kart o una connessione tramite ethernet per attivarla manualmente.

Una volta visualizzati i nomi dei kart nell'apposito spazio sullo schermo, è necessario trascinare quelli che dovranno partecipare alla gara in una delle caselle di partenza, che devono coincidere con quelle reali in cui saranno posizionati sulla pista. Alla pressione del pulsante "*Start race*", il semaforo effettuerà il conto alla rovescia e la gara potrà incominciare allo spegnimento delle tre luci rosse. Da questo momento ha inizio la battaglia che prosegue per quattro intensi giri, seguendo le regole indicate nel *Capitolo 3*. Alla fine della gara, verrà visualizzata sullo schermo la classifica e, tramite il pulsante "*New race*", sarà possibile iniziarne una nuova. Alla conclusione della sfida, il programma in esecuzione sui kart termina, dopo averli disconnessi dall'applicazione centrale. Ciò implica che all'inizio di una nuova gara tale programma debba essere riavviato, come descritto precedentemente.